DSP virgule flottante TMS320C67 MANIP 1
Sur carte "Starter Kit TMS320C6713DSK"

Chaine de traitement numérique du signal et ses défauts associés (échantillonnage, reconstitution. repliement spectre …).
Filtre numérique FIR (programmé en C). Application Moyenne Mobile.

1…5 Rappels théoriques simples de traitement du signal
6 Description chaîne à DSP, maquette TMS320C6713 DSK
7 **Questions et manipulations** sur les défauts inhérents à la chaîne de base.
8 Filtre moyenne mobile FIR (en C). PRATIQUE :
Questions et manipulations. Application **mesure de Veff** d'un signal.

Ne **jamais travailler** sur des programmes **portant un autre nom que celui suggéré** !!! (sinon les outils ne fonctionneraient pas forcément).

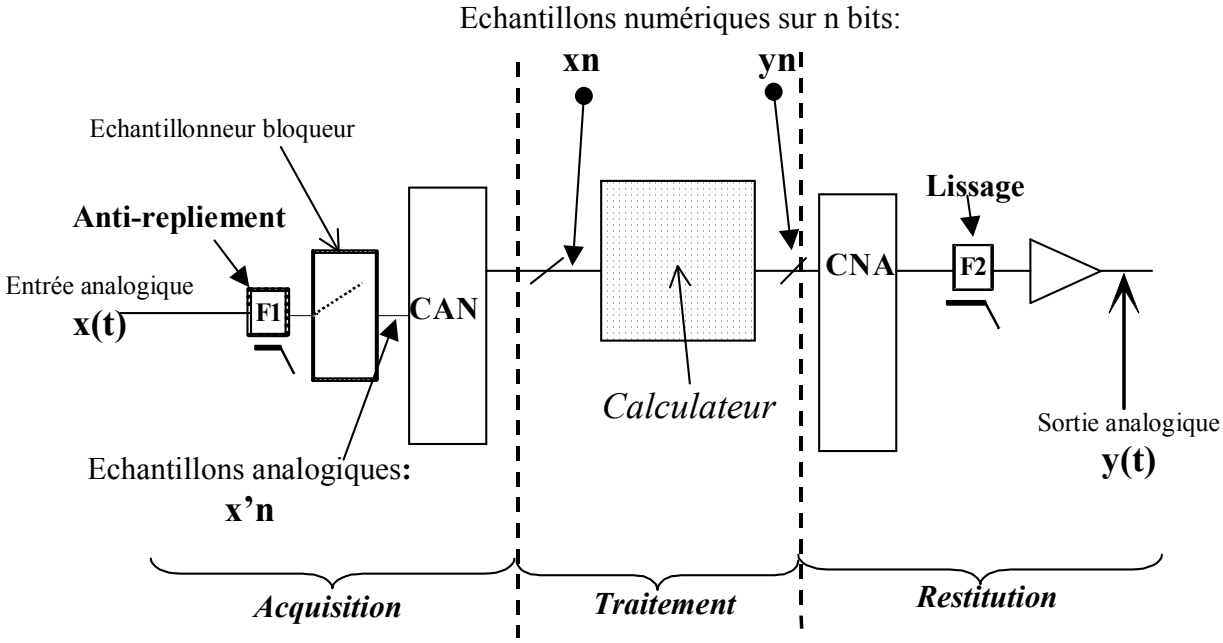
Tous les fichiers vous seront fournis, il vous faudra parfois ajouter quelques lignes de C.

→ Pour ces TP, partie **théorique et pratiques sont assez mélangées**, on pourra rédiger les questions théoriques en laissant de la place pour la pratique.

1. CHAINE DE TRAITEMENT DE SIGNAL EN NUMERIQUE

Le traitement numérique des signaux est une technique dont la théorie et les avantages sont en fait connus depuis longtemps, mais qui n'est devenue vraiment exploitable que depuis peu grâce aux progrès des composants électroniques en puissance de calcul (ordinateurs, processeurs de signaux ……). De nombreux filtres ou traitements (en particulier les filtres à phase linéaire) ne peuvent s'effectuer qu'en numérique (ou du moins qu'au moyen d'échantillons du signal).

1.1. Eléments constitutifs



L'***échantillonneur-bloqueur*** "discrètise" le signal en prélevant des échantillons à une cadence d'échantillonnage **Fech = 1/Tech** et les envoie au CAN (signal **x'n**)

Le ***CAN*** permet la **numérisation** de chaque échantillon et fournit les **xn**.

Le « **Calculateur »** (microcontroleur, ordinateur, processeur de signal, ou circuit câblé …) permet de faire des traitements sur des données numériques.

Un **signal se sortie numérique** peut être fourni sous formes d'échantillons **yn**, à la même cadence Fech de préférence.

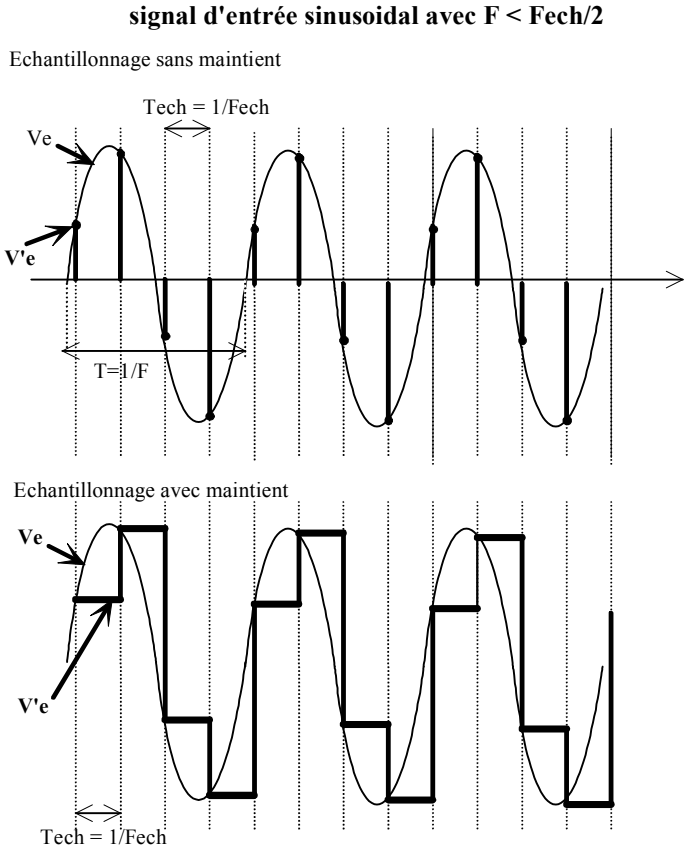
Ces échantillons peuvent restituer un **signal de sortie analogiqu**e par passage par un **CNA** et un **filtre F2** dit de **lissage** (jouant le rôle d'interpolateur).

Le rôle de F1 (filtre anti-repliement) est expliqué plus loin.

1.2. Echantillonnage et reconstitution

1.2.1. Rôle de l'échantillonneur bloqueur

Soit un signal x(t) sinusoïdal à échantillonner, de fréquence **F**. Et soit **Fech** la cadence de prise en compte des échantillons.



Le maintient entre deux échantillons permet:

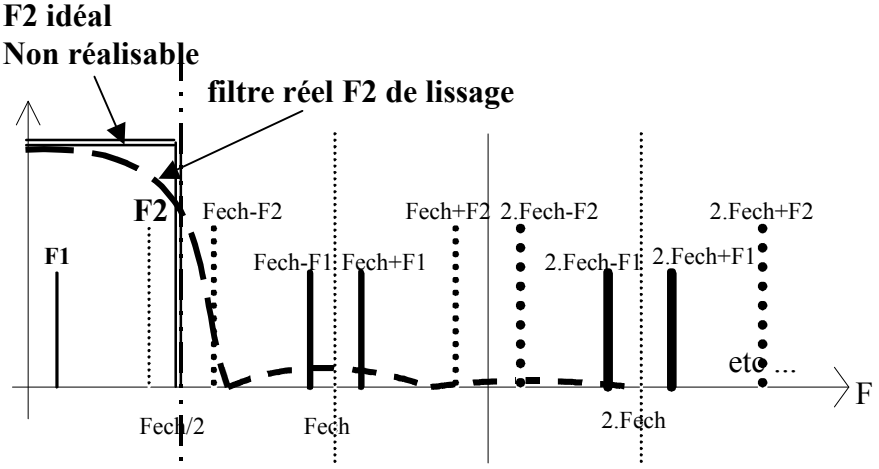
-En **acquisition** pour avoir le **temps de convertir** en binaire les valeurs

-En **restitution** pour avoir un signal **d'énergie suffisante**

1.2.2. Spectre du signal échantillonné, reconstitution

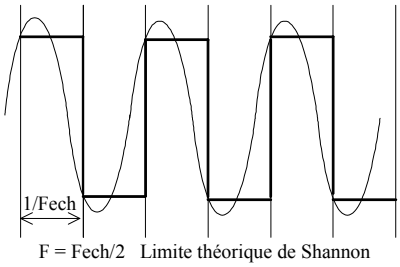
Soit un signal **Ve sinusoïdal**, de **fréquence F**, on montrerait qu'il fournit après échantillonnage toute une série de fréquences supplémentaires **Fech-F Fech+F 2.Fech-F 2Fech+F 3.Fech-F 3.Fech+F .....**

Pour deux fréquences F1 et F2 $< \frac{Fech}{2}$, on aura donc:



Tant que les raies Fech _Fi sont au delà de Fech/2, on voit donc que par suppression des fréquences élevées au delà de Fech/2, on peut reconstituer le signal de départ. Le filtre à utiliser est un **passe bas** dit de **"lissage".**

Cas limite F = Fech/2:

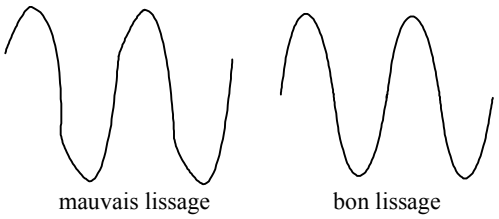


Il faut donc toujours avoir **F < Fech/2** (sauf cas très particulier de "sous échantillonnage"). C'est le **théorème de Shannon**.

➢ ***Le filtre de lissage :***

Un **filtre idéal** laisserait tout passer à Fech /2 et rien au delà, ce n'est évidemment pas possible.

La reconstitution est en fait très facile pour F1, et un peu moins pour F2 qui est plus proche de Fech/2 .



Pour **reconstituer** proprement (avec très peu de distorsions) le signal de départ jusqu'à des fréquences proches de Fech/2, le **filtre de lissage réel** devra donc être **performant :**

- Le plus plat possible ou très peu oscillant **jusqu'à presque Fech/2**

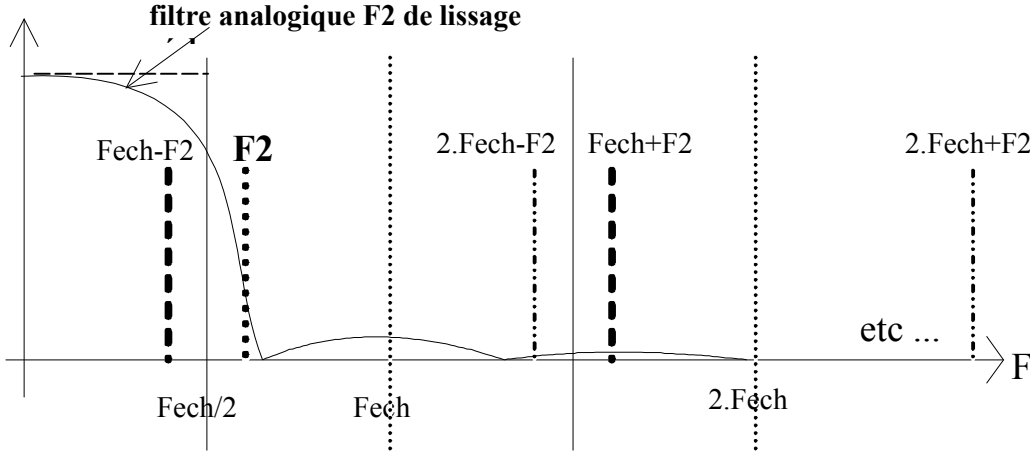
- Atténuer déjà à Fech/2 de 40 à 50dB.

On choisit des filtres classiques analogiques de Butterwoth ou Tchebycheff, souvent d'ordre assez élevé (6 ou 7).

1.2.3. Repliement de spectre

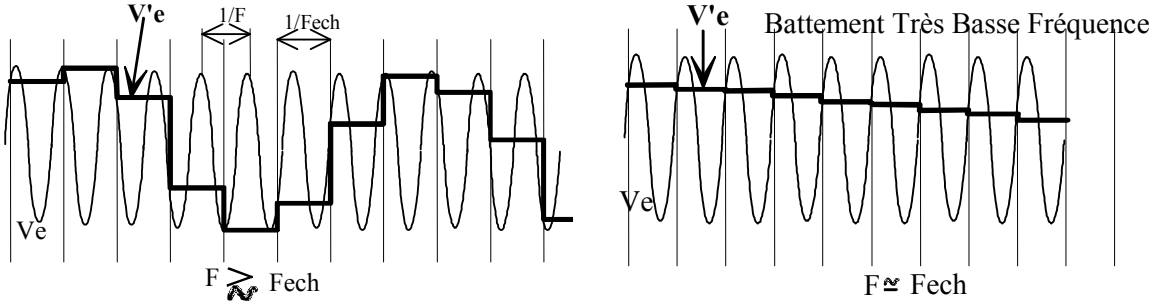
➢ ***Signal Ve sinusoïdal***

Pour une sinusoïde de fréquence F2 à l'entrée et si F2 > Fech /2 :



Le filtrage récupère la fréquence Fech-F2 et non F2 !!

Etude qualitative autour de Fech:



Pour F ≈ Fech, la fréquence de V'e est Fech - F ≈ 0 !!!. C'est un "battement" à très basse fréquence ou à fréquence nulle (si F = Fech), que l'on voit sur la figure ci-dessus.

Ce phénomène se retrouverait autour de 2.Fech de 3.Fech ... de k.Fech !

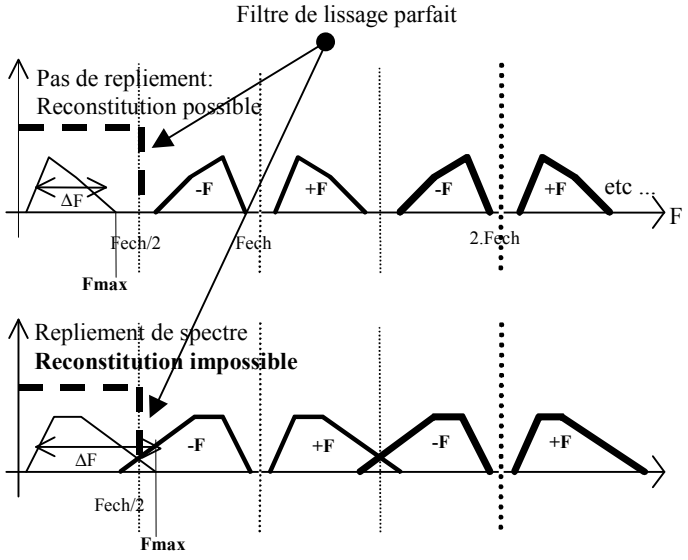
Application du repliement de spectre: Ce phénomène de battement en basse fréquence est mis à profit par les oscilloscopes à échantillonnage (permettant de visualiser des signaux de 50Mhz à plusieurs dizaines de GHz). On l'utilise maintenant également dans certaines techniques de démodulation AM FM en radiocommunication.

On peut citer aussi l'effet stroboscopique qui est en fait un échantillonnage par éclairs lumineux.

- ***Cas d'un signal de largeur de bande ΔF:***

Le **filtre** de **lissage** (**F2** du schéma général), limitant la bande à Fech/2, ne permet de *reconstituer le signal* analogique de départ que *dans le cas ou Fmax < Fech/2.*

Le rôle du **filtre F1** optionnel (du schéma général) est de limiter la bande à Fech/2 ***avant échantillonnage***, afin d'éviter l'apparition de fréquences parasites, on le nomme filtre **"anti repliement"**. Il sert aussi, dans le cas d'un passe bande, à supprimer la composante continue qui doit souvent être nulle en traitement de signal.



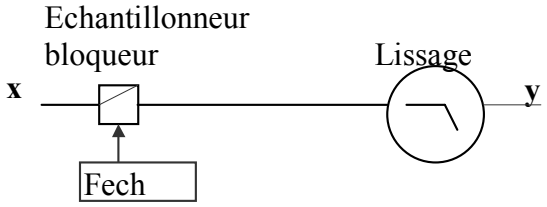
Le signal de la figure du haut à un spectre avec Fmax < Fech/2 donc F1 n'est pas utile.

Le signal de la figure de bas à un spectre dépassant Fech/2, si on l'échantillonne sans filtrage initial, des battements parasites surviendront et ceux ci ne pourront plus être éliminés par la suite, reconstitution impossible !

Ce filtre anti repliement est ainsi indispensable dans de nombreuses applications, comme par exemple avant échantillonnage du signal audio en vue d'enregistrement sur compact disque.

1.2.4. Fonction de transfert en fréquence de l'ensemble échantillonnage et lissage.

Soit la chaîne suivante, avec un signal d'entrée sinusoïdal x = X.cos(2πft)

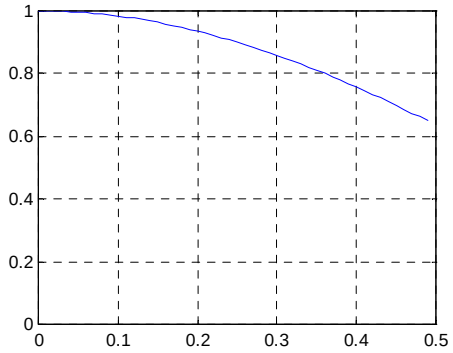


On pourrait montrer facilement (porte de largeur Te) que le bloqueur à lui seul possède une fonction de transfert qui atténue déjà légèrement le signal avant Fech /2 : son module vaudrait en effet :

$\frac{Sin(\pi x)}{\pi x}$ avec x = f/fech.

Ce défaut peut en fait si nécessaire se corriger numériquement. Il n'est pas forcément gênant.

D'autre part la fonction de transfert du filtre réelle du filtre de lissage atténue elle aussi légèrement avant Fech/2.



Une mesure de la simple chaîne acquisition → restitution peut donc s'effectuer, et on peut en tenir compte pour corriger les mesures sur un filtre numérique. Un filtre numérique ne pouvant pas aisément sur notre chaîne s'étudier seul.

1.3. Définitions diverses

1.3.1. Rapport signal sur bruit d'un signal

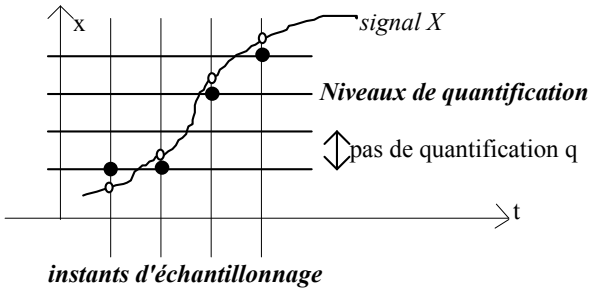
Soit un signal **X** constitué du signal utile **x** et d'un bruit **b**: **X = x + b**

En traitement du signal on parle peu d'erreur maximum ou de précision, mais de rapport signal à bruit exprimé en **dB**, qui est un rapport d'énergie (valeurs efficaces).

Il vaut $S/B = 10.\log\frac{x^2_{eff}}{b^2_{eff}}$ Et si le bruit est faible $S/B \approx 10.\log\frac{X^2_{eff}}{b^2_{eff}}$

Le bruit peut être du bruit seul, mais aussi des harmoniques provenant de distorsions. On parle alors de rapport S/(bruit+distorsion).

1.3.2. Numérisation, bruit de quantification



● = échantillons Xk choisis au plus près par le CAN

o = valeurs exacte X du signal aux instants d'échantillonnag

La différence y = Xk-X constitue une erreur **centrée**, comprise entre -q/2 et +q/2 provenant de la quantification.

On la nomme bruit de quantification.

On pourrait montrer que la valeur efficace de ce bruit s'exprime par $\mathbf{b^2eff = q^2/12}$

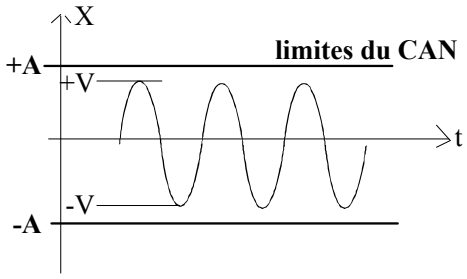
Prenons maintenant une sinusoïde d'amplitude V, échantillonnée, puis convertie par un CAN de n bits, travaillant entre -A et +A.

- ***Raisonnement simple en Erreur Absolue Max sur Xk*** :

Il faut évidemment exploiter le plus possible le CAN (sans toutefois écrêter !), donc V voisin de A.

$$\text{Erreur absolue max} = \pm\, q/2 = \pm\frac{1}{2}\, LSB = \pm\frac{A}{2^n}$$

(LSB = poids du bit de faible poids du CAN).



- ***Raisonnement en Rapport Signal sur Bruit***

Signal : $X^2eff = V^2/2$

Pas de quantification : $q = A/2^{n-1}$ **Bruit :** $y^2eff = q^2/12 = \frac{A^2}{3.2^{2n}}$

Le rapport signal sur bruit vaut donc $S/B = 10.\log (X^2eff / y^2eff)$

$S/B = 10.\log(2^{2n}) + 10.\log(3/2) + 20.\log(V/A)$

Donc : **$S/B = 6{,}02\ n + 1{,}76 - 20.\log(A/V)$** avec **$A >= V$**

On voit, ce qui était déjà évident, que S/B est maximum si on exploite toute l'excursion du CAN. Dans le cas ou **V = A** on a **$S/B = 6{,}02\ n + 1{,}76$**

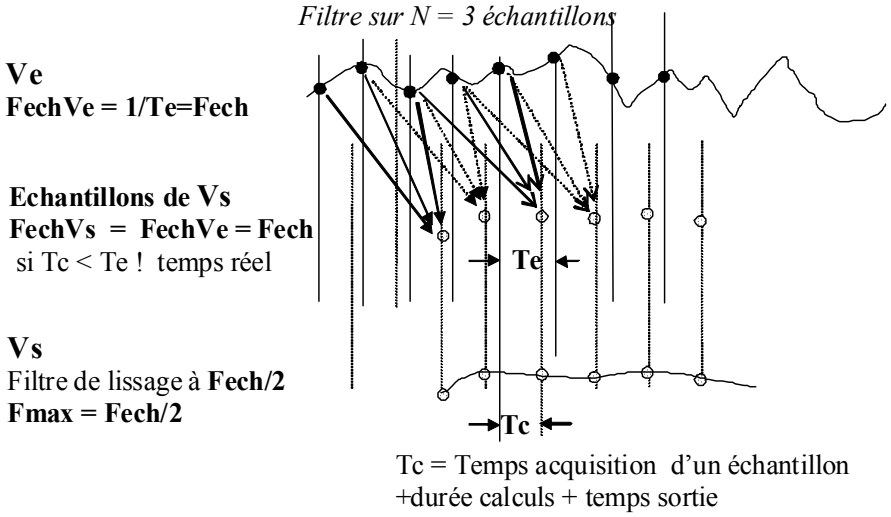
1.3.3. Traitements temps réel ?

➢ **Traitement "en ligne" ou "au fil de l'eau …": vrai filtrage numérique**

Soit Tech la période d'échantillonnage du signal X à traiter. Ce traitement consiste en trois phases: **acquisition** d'un échantillon **Xn**, **traitement** (à partir de Xn, et d'un certain nombre d'échantillons précédents pouvant provenir de différentes sources), **sortie** d'un échantillon **Yn** du signal de sortie Y.

On doit toujours conserver la même cadence d'échantillonnage pour le signal de sortie traité Y (**Fech entrée = Fech sortie),** l'échantillon Yn devra être fourni avant l'acquisition du nouvel échantillon Xn+1. Il faudra donc que

Tacquisition + Tcalcul + Tsortie < Tech.

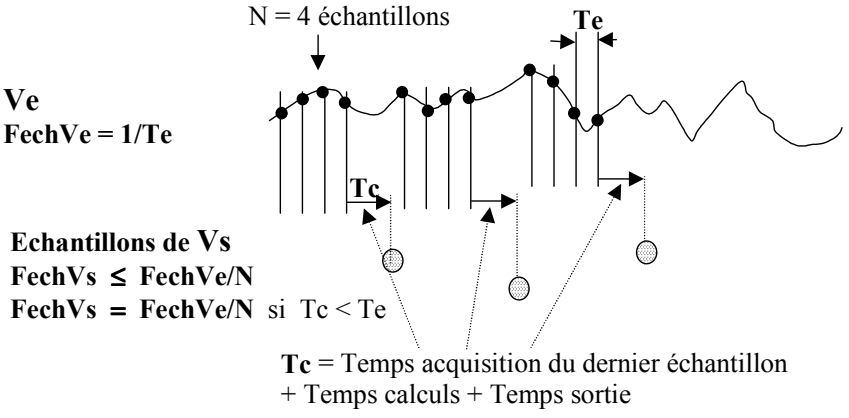


Tc = Temps acquisition d'un échantillon
+durée calculs + temps sortie

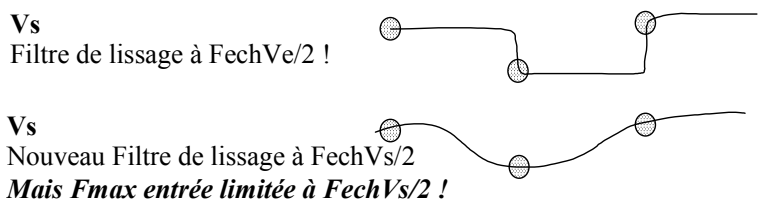
➢ **Traitement par bloc simple**

On fait l'acquisition de N valeurs du signal X. Le traitement s'effectue ensuite.

On travaille parfois ainsi, mais ce n'est pas du filtrage numérique ! Fech de sortie est inférieure à Fech d'entrée.



Tc = Temps acquisition du dernier échantillon
+ Temps calculs + Temps sortie



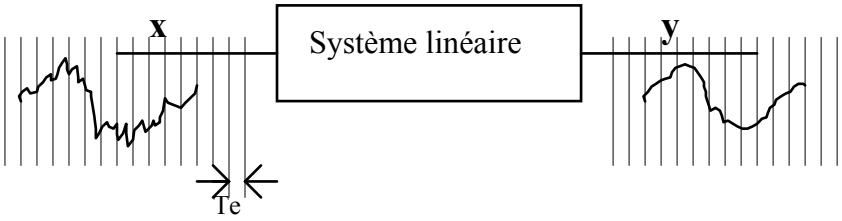
Mais Fmax entrée limitée à FechVs/2 !

➢ **Traitement par bloc Temps réel**

Possible avec une technique à **double buffer**. Voir plus loin.

2. REPONSE IMPULSIONNELLE ET CONVOLUTION

2.1. Système linéaire



instants d'échantillonnage identiques pour tous les signaux: t = nTe = n/Fe

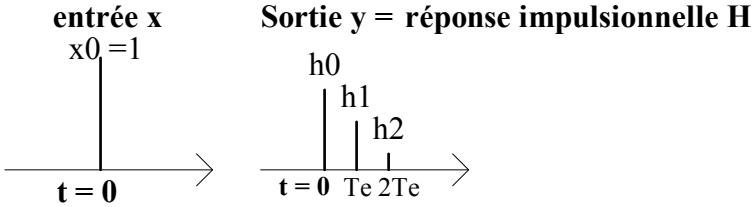
Linéarité: si x = x1+k.x2 alors y = y1 + k.y2
Invariant dans le temps: x(t-τ) donne y(t-τ)
Certains systèmes varient dans le temps, mais si cette variation est lente, on peut les classer dans cette catégorie.
On ne s'intéresse désormais qu'aux valeurs aux instants d'échantillonnage **nTe**.

2.2. Réponse impulsionnelle (d'un système linéaire)

Si l'entrée **x** est une seule impulsion valant 1 pour t = 0, la réponse du système se nomme la réponse impulsionnelle **H** .

Elle peut être de **durée finie** ou **infinie :**
On distingue en effet deux types de filtres :
- **FIR** Finite Impulse Response
- **IIR** Infinite Impulse Response

Exemple simple:
Une réponse de trois échantillons (les suivants éventuellement négligeables)

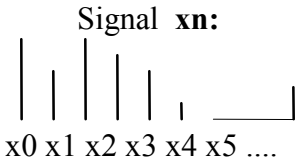


Ce filtre est évidemment « ***à réponse impulsionnelle finie*** **»** ou **« FIR** »

2.3. Construction de la réponse yn à une suite d'échantillons xn. Convolution

Chaque échantillon xk, d'amplitude **xk**, donne une réponse égale à **xk.H**
Toutes ces réponses sont à sommer avec le décalage approprié.

On peut se représenter cela très facilement au moyen de l'exemple précédent qui est un filtre à réponse impulsionnelle finie de N = 3 échantillons:



On a $y0 = x0.h0$

$y1 = x1.h0 + x0.h1$

$y2 = x2.h0 + x1.h1 + x0.h2$

$y3 = x3.h0 + x2.h1 + x1.h2$

$y4 = x4.h0 + x3.h1 + x2.h2$

etc ...

$y_n = x_n.h0 + x_{n-1}.h1 + x_{n-2}.h2$ terme général

Soit **N la taille** du filtre. Pour un filtre à réponse impulsionnelle infinie, l'expression finale sera identique avec N tendant vers l'infini.

A partir de y_3 (indice = **taille N** de la **réponse impulsionnelle H**), on obtient y en sommant toujours N=3 termes.

On voit que la sortie se calcule en permanence à partir de x_n et des N-1 = 2 échantillons précédents, par l'expression:

Cette expression se nomme la **Convolution** de x avec h. On écrit parfois **y = x * h** (produit de convolution).

$$y_n = \sum_{k=0}^{N-1} x_{n-k} . h_k$$

Conclusion :

A l'arrivée d'un signal x_n, un **régime transitoire** dure donc exactement **NTe** soit la **durée de la réponse impulsionnelle**.

A tout instant, la sortie est fonction de la nouvelle entrée x_n et de N-1 valeurs de l'entrée précédente. Donc prédictive exactement.

Pour un **filtre IIR**, N est infini, le régime transitoire est 'théoriquement' infini, mais heureusement pas en pratique ! c'est un peu comme une exponentielle en analogique qui bien que théoriquement infinie, ne varie pratiquement plus au bout de 4 à 5 constantes de temps !

3. REPONSE EN FREQUENCE : FONCTION DE TRANSFERT

On trouvera ici une étude simple permettant de comprendre sans faire appel à des calculs compliqués de transformée en z. La notation **z** sera seulement introduite pour simplifier l'écriture.

3.1. Etude générale avec x sinusoïdale, filtre de réponse impulsionnelle H

x = A.cos(2πft)

xn = A.cos(2πnf/Fe) ou **xn = A.cos(2πn.fTe)**

On pose x = fréquence normalisée: **x = f/Fe = fTe** sans dimension.

xn = Acos(2πnx)

0 ≤ x ≤ 0.5
0 ≤ f ≤ Fech/2

Comme en analogique, pour calculer une fonction de transfert **G** qui apportera atténuation et déphasage en fonction de f, on introduit la notation complexe:

Soit $Xn = Ae^{2\pi inx}$ avec xn = Reel(Xn)

Alors : $$G(x) = \frac{Yn}{Xn} = |G(x)|.e^{i.\varphi(x)}$$

G(x) est la **fonction de transfert complexe** du système

Le module du gain |G(x)| peut s'exprimer en dB: $G_{dB}=20.\log(|G(x)|)$

Le déphasage est $\varphi(x)$

Pour une réponse impulsionnelle H, on obtient la sortie par la convolution : $Y_n = \sum_{k=0}^{N-1} X_{n-k}.h_k$

Or : $X_{n-k} = Xe^{2\pi inx}.e^{-2\pi ikx}$

Donc Xn-k signifie **donc déphasage** de **–2πkx** = -2πkfTe

Rappel : Si une sinusoïde sin2πft est retardée de τ, elle s'écrit : sin2πf(t-τ) = sin(2πft – 2πfτ) = sin(2πft-ϕ)

Un **retard τ** provoque donc un **déphasage** de **–2πf.τ**

Donc Xn-k signifie **donc un retard** de **k.Te**

Il vient $$Y_n = \sum_{k=0}^{N-1} X_{n-k}.h_k = \sum_{k=0}^{N-1} Xe^{2\pi inx}e^{-2\pi ikx}.h_k = Xn\sum_{k=0}^{N-1} e^{-2\pi ikx}.h_k$$

D'ou $$G(x) = \frac{Yn}{Xn} = \sum_{k=0}^{N-1} e^{-2\pi i k x} . h_k$$

On pose souvent (ici pour simplifier l'écriture et aussi pour une étude plus générale dite transformée en z) : $Z = e^{2\pi i x}$

Et alors : $$G(x) = \frac{Yn}{Xn} = \sum_{k=0}^{N-1} z^{-k} . h_k$$ avec $$z = e^{2\pi i x}$$

3.2. Gain pour f = 0 (donc pour le continu)

Valable donc seulement pour un passe bas !

On fait f = 0, donc **x = 0** ou **z = 1**, il vient immédiatement : $$G_0 = \sum_{k=0}^{N-1} h_k$$

Pour avoir donc un gain continu ou très basse fréquences, de **1**, il faut que la somme des coefficients soit égale à **1**

3.3. Valeur maximale de |G| pour tout filtre :

$$G_0 = \sum_{k=0}^{N-1} |h_k|$$

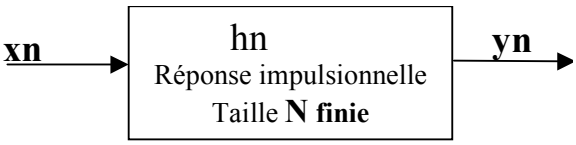
Pratique pour étudier tout débordement.

4. FILTRES FIR : NON RECURSIFS REPONSE IMPULSIONNELLE FINIE

4.1. Description

- Décrits par l'équation :

$$y_n = \sum_{k=0}^{N-1} x_{n-k}.h_k$$



Nommés filtre **Non récursifs**
Evidemment **toujours stables**

- Fonction de transfert en fonction de la fréquence:

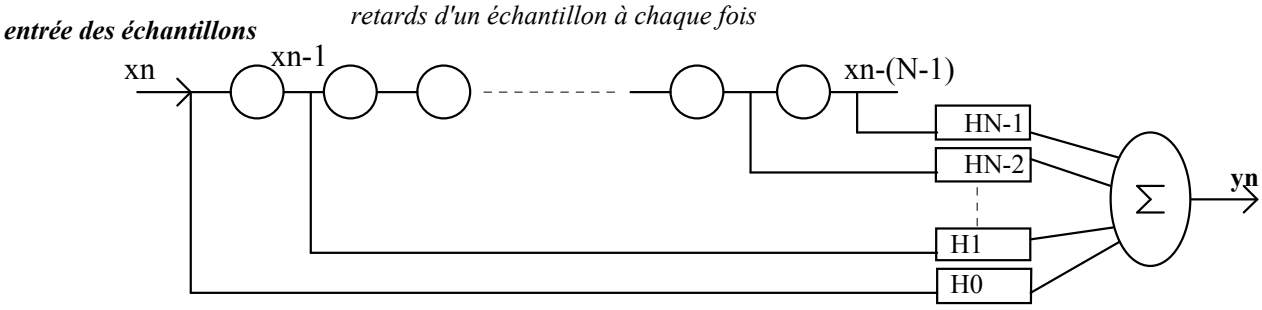
$$G(x) = \frac{Yn}{Xn} = \sum_{k=0}^{N-1} e^{-2\pi i k x}.h_k \quad \text{avec} \quad x = f / Fech = f.Te$$

Ecriture rapide en Z :

$$G(x) = \frac{Yn}{Xn} = \sum_{k=0}^{N-1} z^{-k}.h_k \qquad z = e^{2\pi i x}$$

- Réalisation :

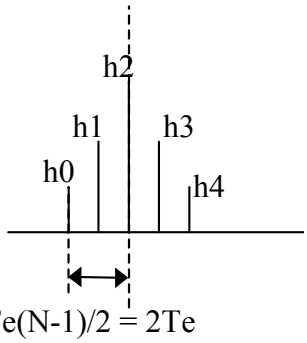
La structure (qui peut être câblée ou programmée) réalisant ce filtrage se nomme **filtre "transversal"** comme le suggère la figure suivante. Il est évident que si on présente à l'entrée un seul échantillon à 1, on récupère en sortie la suite des coefficients et donc la réponse impulsionnelle h.



4.2. Filtres à réponse impulsionnelle finie et filtres à phase linéaire ?

On veut toujours réaliser des filtres ne déformant pas le signal dans la bande utile. Or on sait qu'un filtre (comme une ligne de transmission) provoque des distorsions non désirées si la phase n'est pas linéaire, les différentes fréquences constituant le signal ne se propageant alors pas à la même vitesses (dispersion). Tous les filtres analogiques sont par nature dispersifs.

La condition pour limiter cela est de s'approcher le plus possible d'un déphasage proportionnel à la fréquence dans toute la bande utile.



- ***Réponse impulsionnelle*** de taille impaire (l'étude est plus aisée) et ***<u>Symétrique</u>*** :

Ici taille N = 5

$$G = \sum_{0}^{N-1} e^{-2\pi ikx}.h_k = h_0 + h_1.e^{-2\pi ix} + h2e^{-4\pi ix} + h3e^{-6\pi ix} + h4e^{-8\pi ix}$$

En mettant $e^{-4\pi ix}$ en facteur et en regroupant deux à deux les hi symétriques, il vient :

$$G = e^{-4\pi ix}\left(\frac{h0+h4}{2}\cos 4\pi x + \frac{h1+h3}{2}\cos 2\pi x + h2\right)$$

Le premier terme est un déphasage de $4\pi x = 4\pi fTe$, et donc un **retard pur** de 2Te

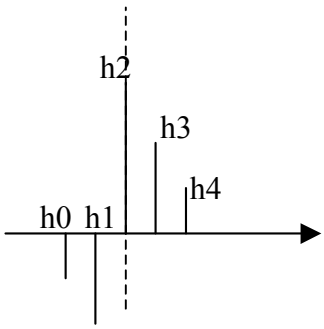
Le second est réel.

Un filtre à **réponse impulsionnelle finie** et **symétrique** permet donc de réaliser des **filtres parfaitement à phase linéaire** (et donc sans distorsion due à des temps de propagation différent selon la fréquence), comprenant :

- Un **gain réel**
- Un **retard pur θ (**obligatoirement mais ce qui n'est nullement un inconvénient …) égal à la <u>moitié</u> de la durée de la <u>réponse impulsionnelle</u>, soit $\theta = \frac{N-1}{2}Te$

C'est le **grand intérêt** des **filtres** numériques à **réponse impulsionnelle finie** !

Un tel filtre n'as pas d'équivalent en analogique !



- ***Réponse impulsionnelle Anti-<u>symétrique</u>*** :

Ici taille N = 5

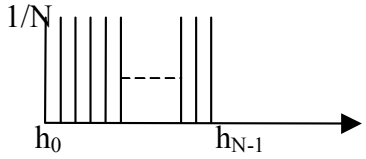
On retrouverait la même propriété mais un déphase supplémentaire de $\pi/4$ (termes en sinus) qui peut servir dans certaines applications.

5. LE FILTRE FIR « MOYENNEUR » (NON RECURSIF)

C'est un filtre tout bête et très classique dont les applications sont, nous le verrons multiples selon ses paramètres.

5.1. réponse impulsionnelle

Pour un filtre de **taille N**, tous les **hk = 1/N**



La convolution $y_n = \sum_{k=0}^{N-1} x_{n-k}.h_k$ donne alors $y_n = \frac{1}{N}\sum_{k=0}^{N-1} x_{n-k}$ qui est évidemment la moyenne des N valeurs de $x_{n-(N+1)}$ à x_n d'où le nom de **filtre moyenneur.**

⇒ ***Différence avec une simple moyenne :***
Sans parler de convolution, on sait ce qu'est évidemment une simple moyenne ! on peut en effet acquérir N valeurs d'un signal et calculer ensuite la moyenne. La différence réside ici dans le fait que l'on dispose en permanence de la moyenne de l'échantillon présent et des N-1 précédents, d'ou le nom de « **moyenne mobile** », et c'est alors un vrai filtre.

5.2. Calcul de la fonction de transfert

Par application de la formule de la somme des N premiers termes d'une progression géométrique $u_n = aq^n$: $\sum_{k=0}^{N-1} aq^k = a\frac{1-q^N}{1-q}$ on obtient :

$$G = \sum_{k=0}^{N-1} e^{-2\pi ikx}.h_k = \frac{1}{N}\frac{1-e^{-2\pi iNx}}{1-e^{-2\pi ix}} = \frac{1}{N}\frac{e^{i\pi x}}{e^{i\pi Nx}}\frac{e^{i\pi Nx}-e^{-i\pi Nx}}{e^{i\pi x}-e^{-i\pi x}} = \frac{1}{N}\frac{e^{i\pi x}}{e^{i\pi Nx}}.\frac{\sin \pi Nx}{\sin \pi x}$$

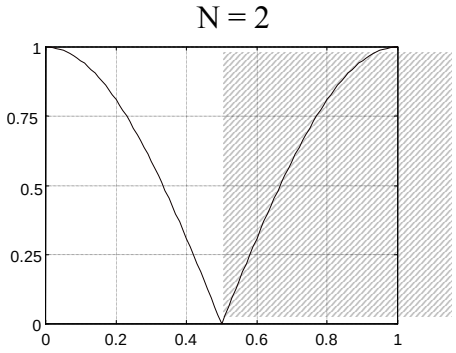
On ne s'intéresse ici qu'au **module de G (qui représente un gain ou un affaiblissement)**

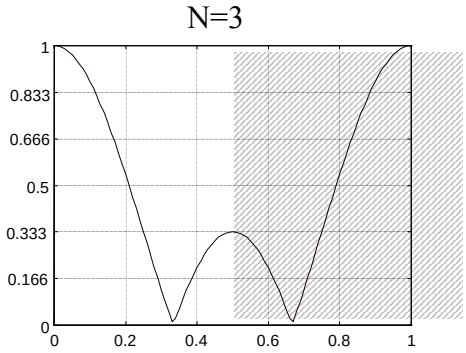
$$\text{Donc } |G| = \frac{1}{N}.\left|\frac{\sin \pi Nx}{\sin \pi x}\right|$$

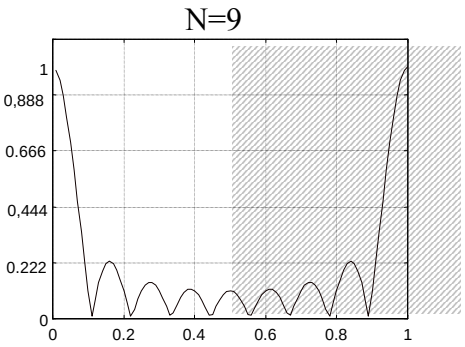
Traçons pour différentes valeurs de N, le module de la fonction de transfert due à ce filtre.

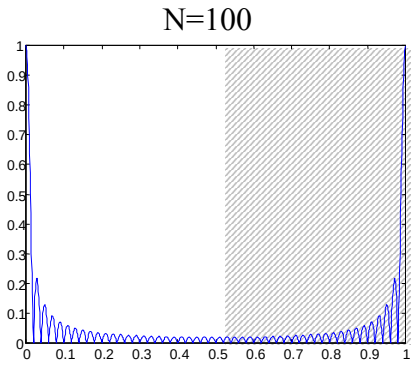
Remarque très importante : au delà de Fech/2, on observe mathématiquement une sorte de fonction de transfert « miroir ». Il faut se souvenir que l'on ne peut parler de Fonction de transfert que si la fréquence d'entrée est identique à celle de sortie (Système linéaire), donc seulement pour **F ≤ Fech/2**. Au delà de Fech/2, Shannon n'est plus respecté, il y a repliement de spectre. La sortie n'est plus de fréquence F, mais Fech-F, puis 2Fech-F ….

Les **zéros de |G|** correspondent à $\pi Nx = k\pi$ donc à x = k/N ou $\boxed{f_{zeros} = k.\frac{f_{ech}}{N}}$ **k ≠ 0**









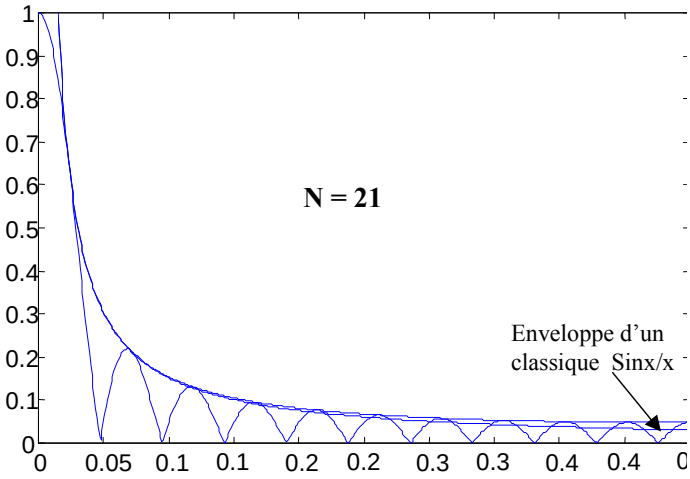
Les **maximums** correspondent presque au numérateur sinπNx = 1.

D'ou l'équation approchée de la courbe des maximums :

$$\boxed{Max(x) \approx \frac{1}{N}.\left|\frac{1}{\sin \pi x}\right|}$$

La fonction de transfert est très proche d'un |sinx/x| classique, surtout au début.

Exemple pour un filtre à N = 21 coefficients :



Utilisation :

C'est donc un **filtre passe bas**, qui atténue les fréquences élevées et provoque donc un **lissage** plus ou moins prononcé du signal.

Il Permet de calculer une **valeur moyenne**, une **valeur efficace** (si on l'applique sur les carrés des échantillons).

On s'en sert aussi couramment dans d'autres domaines, par exemple pour lisser les courses de bourse (avec des moyennes mobiles sur 1 semaine, 1 mois ou plus, et trouver certaines tendances des marchés).

5.3. Application N°1 : Filtre Moyenne Mobile

5.3.1. Fonction de transfert

C'est d'après l' étude précédente un filtre passe bas, il atténue les fréquences élevées et provoque un lissage plus ou moins prononcé du signal.

On s'en sert couramment dans d'autres domaines, par exemple pour lisser les cours de bourses (avec mes moyennes mobiles sur 1 semaine, 1 mois ou plus), et trouver ainsi des tendances des marchés.

Pour une même Fech, **plus on filtre**, **plus le premier zéro** (et les suivants) sont en **BF**. **Plus** le nombre **N** de coefficients doit être **élevé.**

5.3.2. Régime transitoire

On sait que la durée de ce régime est égal à la durée de la réponse impulsionnelle, soit N.Tech

Donc pour une même Fech, **plus on filtre** en BF et **plus le régime transitoire** est **long**.

5.4. Application N°2: Valeur moyenne d'un signal

5.4.1. Définition mathématique

$$\overline{X} = \frac{1}{T}\int_0^T X(t).dt \quad \text{avec} \quad T \to \infty$$

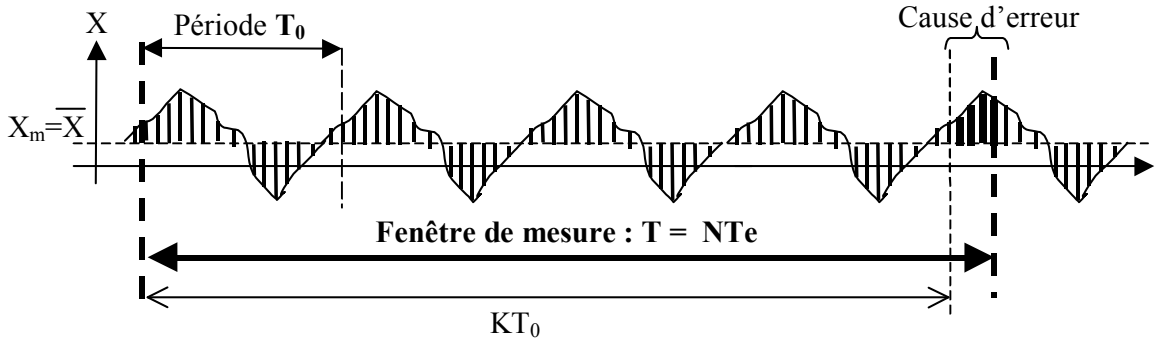
Approche en numérique :

$$\overline{X} = \frac{1}{N}\sum_{k=0}^{N} Xn \quad \text{avec N grand.}$$

5.4.2. Conditions d'une bonne mesure de valeur moyenne

1) Interprétation de manière intuitive sans parler de spectre

→ La fenêtre de mesure T doit avoir une durée multiple de la période T_0 du signal, ou sinon grande par rapport à celle ci.



Sur cette figure, Il est clair qu'un calcul sur une durée égale à T_0, ou $2T_0$ ou kT_0 (durée multiple de la période du signal) fournit une bonne valeur moyenne, mais c'est un cas particulier, car le plus souvent on effectue une mesure sans connaître la fréquence.

La mesure générale sur une durée quelconque T = N.Te fournit toujours une erreur mais d'autant plus faible que cette durée est grande par rapport à la période.

Conclusion pour une **bonne mesure**, et donc une **mesure répétitive stable** :

→ **Cas particulier sans erreur:**

T = N.Te = k.T0, donc **F0 = k.Fech/N** Donc **Mesure** sur **k périodes** de base

→ **Cas général :**

T = NTe >> T0, donc **F0 >> Fech/N** **Mesure** sur une **durée >> période**

2) Interprétation plus théorique par le spectre de fréquence

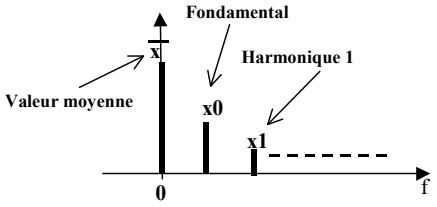
Soit un signal quelconque périodique, sa décomposition en série de Fourier donne :

$$X(t) = \overline{X} + x_0 \cos(2\pi F_0 t + \varphi_0) + x_1 \cos(2\pi F_1 t + \varphi_1) + x_2 \cos(2\pi F_2 t + \varphi_2) + ...$$

Le spectre est donc :

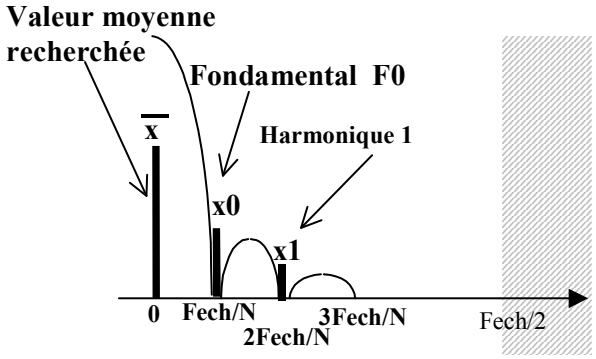
On désire récupérer la valeur moyenne, donc seulement la raie de fréquence 0.

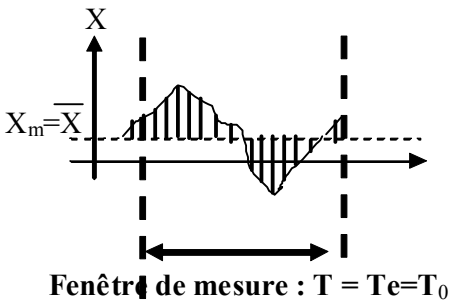
Un **filtre passe bas de très basse fréquence** rempli donc cet office.



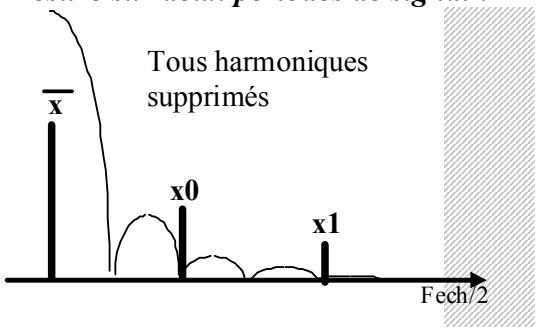
→ **Mesure Parfaite, mais cas particulier ou F0 = k.Fech/N :**

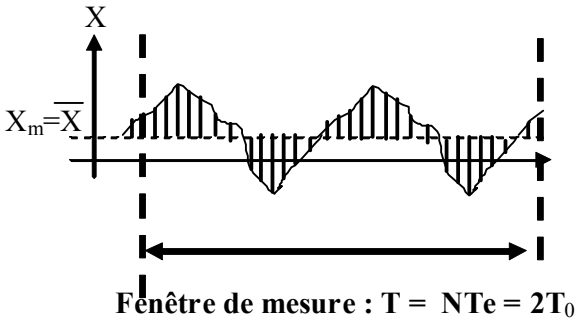
Mesure sur une période de signal :





Mesure sur deux périodes de signal :

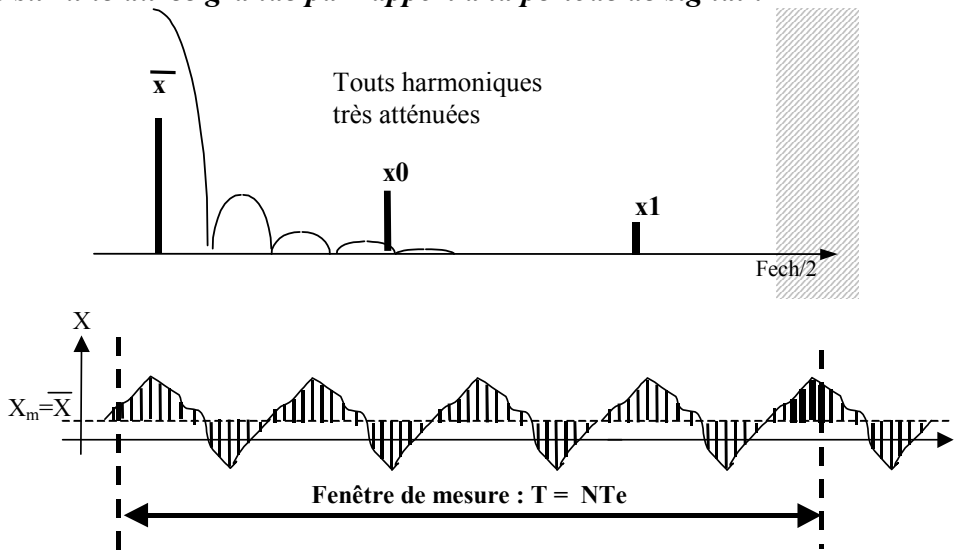




La mesure sera parfaitement répétitive, avec le même résultat à chaque fois.

→ Bonne mesure et cas général: F0 >> Fech/N

Mesure sur une durée grande par rapport à la période de signal :

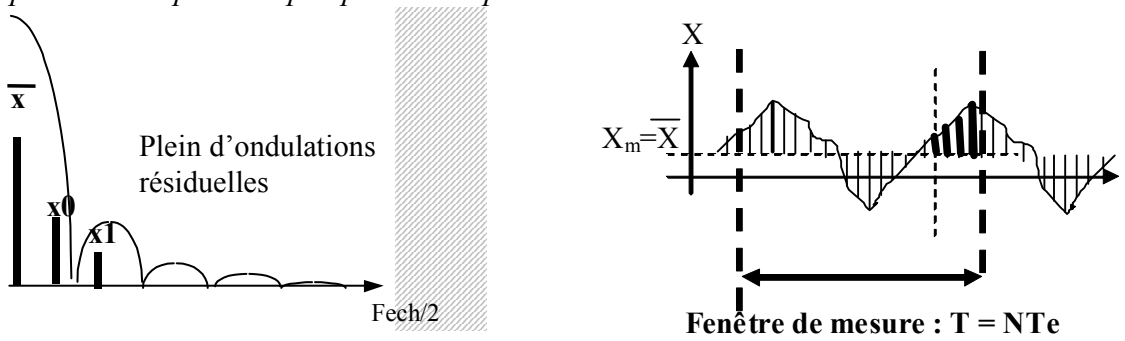


La cause d'erreur en plus ou en moins est la partie grisée, qui a donc de moins en moins d'influence si la durée de mesure augmente.

La mesure sera pratiquement répétitive, avec peu de variation de l'une sur l'autre.
L'erreur maximale pourra se calculer.

→ Cas de mauvaise mesure !

Ici prise en compte d'un peu plus d'une période !

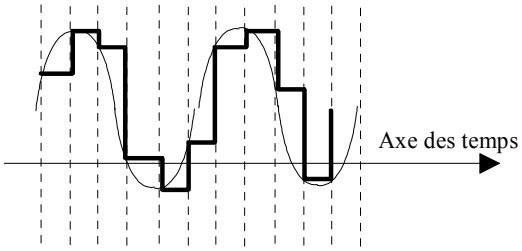


Ici la valeur calculée fluctuera fortement d'une mesure à l'autre !

→ Pour tous les cas : Ne pas oublier de prévoir le **filtre anti-repliement** interdisant toutes fréquences au delà de Fech/2, sinon des fréquences telles que F0 >> Fech/N normalement correctement filtrées pourraient se replier en basse fréquence et provoquer alors de nouveau des ondulations !

3) Condition supplémentaire à ne pas oublier : le nombre d'échantillons sur une période doit tout de même être élevé.

Comme le montre en effet le croquis ci-contre, les surfaces entre la sinusoïde ou la courbe en gras, et l'axe des temps sont égales seulement si N est grand. (Erreur classique lors des calculs de surfaces en numérique, des méthodes permettent des améliorations).



4) Conclusion

En pratique pour augmenter la précision de mesure, on prend souvent une durée supérieure à 100 ou 1000 périodes.

En BF un nombre de plus en plus grand d'échantillons est nécessaire si on ne diminue pas Fech.

Remarque : Pour mesurer une **valeur efficace**, on ferait le même raisonnement puisqu'il faut alors calculer la **moyenne des carrés** (ce n'est plus un filtre linéaire à partir des xn, les échantillons étant élevés au carré).

6. CHAINE DE TRAITEMENT DE SIGNAL A DSP SUR STARTER KIT TMS320C6713 DSK

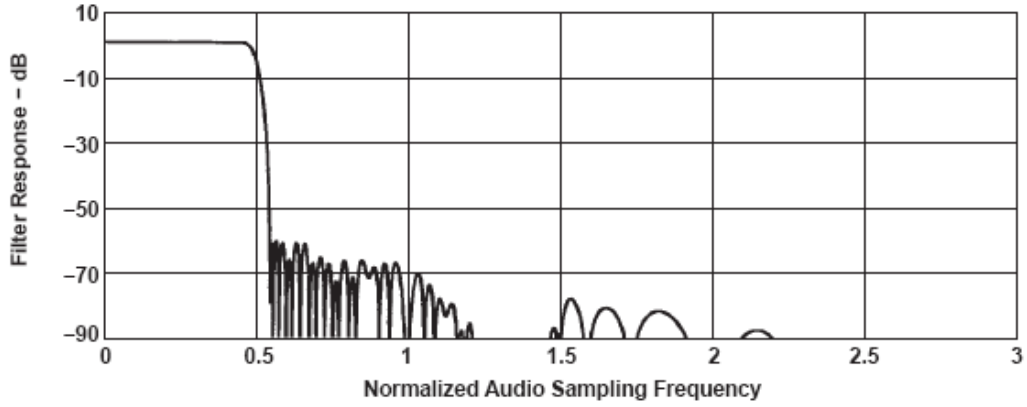
6.1. Description succincte de la maquette

- Elle contient toute la chaîne décrite au début :
  - Entrée signal analogique **deux voies.**
  - Filtre **anti-repliement F1 ici toujours présent** (possibilité de le supprimer sur d'autres cartes).

  - ***Caractéristique de l'interface analogique seul :***

Fs = frequency sample = fréquence d'échantillonnage **Fech**.

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
ADC Filter Characteristics (TI DSP 250 f_S Mode Operation)					
Passband	±0.05 dB	0.416 f_S			Hz
Stopband	−6 dB		0.5 f_S		Hz
Passband ripple				±0.05	dB
Stopband attenuation	f > 0.584 f_S		−60		dB



- Echantillonnage à **Fech = 48 kHz (pour ces TP).**

- Conversion par **CAN**, **deux voies** simultanées. **CNA deux voies** simultanées**.**

- Traitement par le DSP texas : **TMS320C67**, **Fck = 225 MHz**

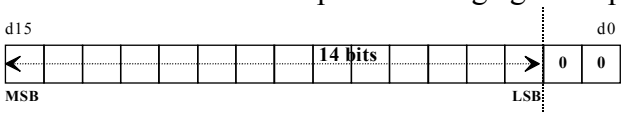
- Filtre **de lissage F2,** caractéristiques voisines de F1, sortie analogique **deux voies.**

- **ATTENTION : Ces cartes** possèdent un **passe haut supplémentaire** en **entrée** et en **sortie** commençant à couper en dessous de 30 Hz (simple capa de liaison).

- **Interfaces Analogiques :**

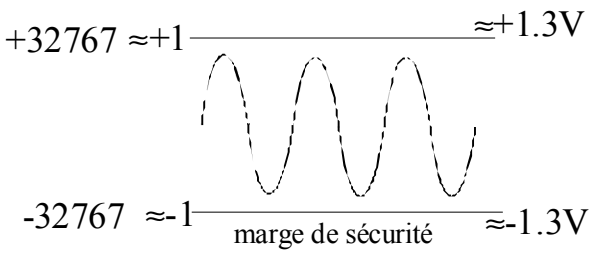
Fréquence d'échantillonnage programmable, et positionnement du filtre de lissage automatiquement à Fech/2.

Convertisseurs **n bits (n <=16).** Données **cadrées à gauche**, donc de dynamique presque identique quel que soit ce nombre de bits. Exemple de cadrage gauche pour 14 bits:



Tensions analogiques :

Entrée **x(t)** et sortie **y(t)**: **max**imum **1,3 volts crête**



Echantillons **xn** et **yn** sur **16 bits signés (mode complément à 2)**, donc en n'utilisant pas –32768, dynamique de :

- De **0 à ±32767** en raisonnement entier
- Ou de **0 à presque ±1** en raisonnement 15 bits fractionnaires : Q15(16)

-,--- ---- ---- ----

→ Eviter de dépasser en entrée et en sortie les valeurs maximales, sinon des distorsions plus ou moins brutales non étudiées ici interviendraient (repliement brutal d'amplitude, écrêtage).

➢ ***Le processeur est un DSP virgule flottante :***

Ce composant est très rapide même en travaillant sur des flottant, son unité arithmétique interne étant conçue pour travailler directement sur une mantisse et un exposant.
Un DSP virgule fixe (calcul en réalité tout en entier) pourrait certes travailler aussi sur des flottants, mais au moyen de librairies de calcul et donc demanderait 10 à 100 fois plus de temps par opérations !

→ Il n'y a pratiquement pas de problèmes de débordement en cours de calcul sur un DSP virgule flottante, mais quelques précautions subsistent tout de même :

→ Si le DSP doit fournir des valeurs à un CNA câblé comme le CAN d'entrée en cadrage gauche, on ne doit pas envoyer d'échantillons plus grands que –32767 et + 32767, sinon la conversion en entier est tronquée provoquant ainsi des <u>débordements</u> et des <u>signaux d'apparence n'importe quoi</u> !

→ ***Conclusion :***

On connaît la dynamique du signal d'entrée x.
Il faut faire attention seulement la **dynamique pratique du signal y fourni au CNA,** pour cela :

Prévoir le <u>gain maximum</u> de la chaîne en <u>fonction de la fréquence.</u>
<u>Prévoir aussi les dépassements lors d'un transitoire</u>.

(En effet un filtre peut très bien avoir un module de gain toujours inférieur à 1 présenter un transitoire un peu oscillant et dépassant ce gain de 1).
Il faut parfois ne sortir au CNA que la moitié, le quart (ou même moins) du signal y calculé.

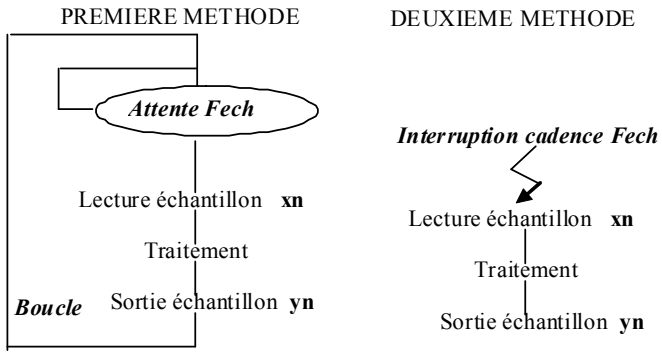
6.2. Organigramme général d'un traitement en temps réel simple

Soit **Fech** et **Tech** respectivement la fréquence et la période d'échantillonnage.

Pour assurer le temps réel, toute la boucle (ou l'interruption) doit avoir une durée inférieure à la période d'échantillonnage **Tech=1/Fech**

Donc, si **Tboucle** est le temps de boucle (ou bien celui du programme d'interruption), il faut que :

Tboucle < Tech



6.3. Variante pour cette maquette et le logiciel fourni

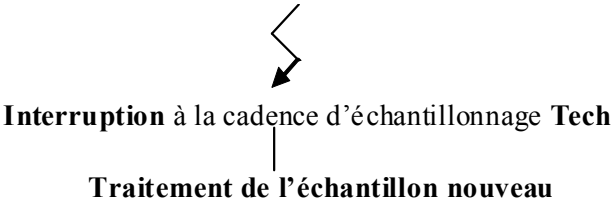
Le fonctionnement réel est assez différent (travail par bloc, avec deux buffers).

Il importe en réalité peu dans ce TP, car on se trouve dans un mode qui ressemble globalement vu de l'extérieur à la deuxième méthode, avec les **entrées** et **sorties** des échantillons **en tache de fond**, et donc non visible dans l'organigramme de votre programme.

Il suffira juste de respecter les notations :

I[0] ou **I[1]** nouvel échantillon entrant (2 voies).
O[0] ou **O[1]** nouvel échantillon (2 voies) qui sera envoyé au CNA.

Organigramme Pratiquement équivalent



La fonction C sur laquelle vous travaillerez se nommera :

void traitement_echantillons(int *I, int *O) ;

7. TRAVAIL THEORIQUE ET PRATIQUE DEFAUTS INHERENTS A LA CHAINE DE BASE

7.1. Projet et programme de démonstration fourni

→ Dans le **répertoire** que vous fournira l'enseignant,

Ouvrir avec le **« Code Composeur Studio »** (qui est l'outil de développement fourni par Texas) le projet : **dsk_c67.pjt**

Aller dans **sources**, il **vous faudra** les fichiers C principaux (et seulement ceux ci) :

dsk_c67.c le noyau de l'application, ne rien modifier !

aic23.c fonctions de l'interface analogique, ne rien modifier

demo.c **Le fichier C de travail** (si ce fichier n'est pas présent dans le projet, ou si il y en a un autre, l'ajouter et ne garder comme fichiers.C dans le projet que ces trois fichiers. Ne rien modifier d'autre évidemment !

Fichier **demo.c** : Le traitement est ici tout simple. On ne fait qu'acquérir un signal à la fréquence Fech, le numériser, et le reconstituer.

```
//                          PROJET toujours dsk_c6713.pjt
// kit de développement TEXAS TMS 320C6713 DSK
// I[0] O[0]: voie gauche ---> "tip" ;  fiche RCA blanche ou noire
// I[1] O[1]: voie DROITE ---> "ring" ; fiche RCA ROUGE
// Dynamique des échantillons:
//      +- 1,3 Volt crète
//      -32767 à +32767    en Q0(16)        (sur des int de 32 bits)
//      presque (-1 à +1)  en Q15(16)  ( Q31(32) en fait )
//Si on veut travailler sur des flottants de -1 à +1,
//                  On divisera par 32768.0  en entrée
//                  On re multipliera par 32768.0 avant de sortir
// //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
#include <log.h>
#include <math.h>
extern far LOG_Obj trace;   // non utilisée ici
void pulse_gpio2(void);   // pour mesurer des durées en temps réel

// Initialisations avant la boucle principale de travail
Init_traitement()       // rien ici
{
}

// Traitement des échantillons comme une simple acquisition un par un (2 voies, stéreo)
void traitement_echantillons(int *I, int *O)        // [0] voie 0     [1] voie 1
{
O[1] =  I[1];          // on travaille ici sur la voie 1    (pourquoi ?....    pourquoi pas la 1 ! ).
}
void periodic_log(void)                // non utilisé ici
{ }
```

→ On remarque en fait deux fonctions que l'on pourra modifier :

Init_traitement() Qui servira pour initialiser des valeurs avant le traitement.

void traitement_echantillons(int *X, int *Y) La fonction de traitement correspondant à votre application

Avec :

I[0] et I[1] Nouvel échantillon entrant (2 voies).

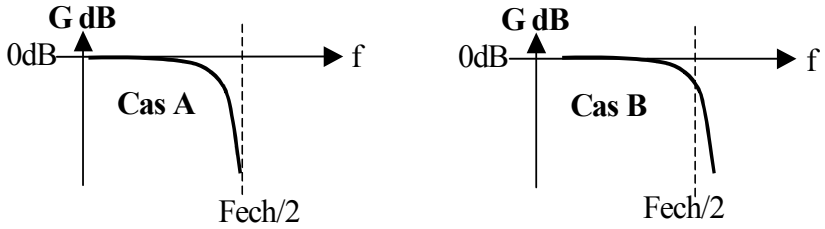
O[0] et O[1] Nouvel échantillon (2 voies) qui sera envoyé au CNA

→ **Ne cherchez pas de programme main() !** il se trouve en réalité dans un autre fichier (dsk_c76.c).

7.2. Positions des filtres F1 et F2 et questions théoriques

→ Les filtres F1 et F2 ont une raideur et une position fixe (par rapport à Fech/2) déterminées par le constructeur.

Mais il y aurait en fait deux choix possibles pour le positionnement de ces filtres:



Cas A : Forte atténuation à Fech/2. On privilégie une bonne protection contre le repliement de spectre (filtre F1), et un bon lissage (filtre F2). Mais en contrepartie la bande passante est réduite.

Cas B : Début d'atténuation à Fech/2. On privilégie la bande passante, en tolérant des fréquences parasites quand on s'approche de Fech/2, et un moins bon lissage.

→ Remarque, en **Haute Fidélité** (HIFI) on est souvent dans le **cas B**, car en fait les fréquences parasites présentes sont inaudibles, et la bande passante est ainsi améliorée pour une même Fech.

D'après les caractéristiques des filtres fournies précédemment :

1) Etes-vous dans le cas A ou dans le cas B, et que privilégie-t-on (meilleur lissage ou meilleure bande passante) ?

2) Donner la fréquence **Fmax = $F_{-0.05db}$** où l'atténuation reste encore négligeable.

3) Donner la fréquence **F_{-6db}** ou l'affaiblissement est de 6dB. Quelle est la valeur du **module** de la fonction de transfert pour cet affaiblissement (cette valeur servira dans la partie pratique pour mesurer en ce point).

4) Donner la fréquence **F_{-60db}** au delà de laquelle l'affaiblissement est > 60dB

7.3. Travail pratique

Eviter toujours de déformer les signaux en ne mettant pas trop d'amplitude de signal !

De plus, si vous aviez multipliez par par exemple 1.5 vos échantillons, la valeur max de sortie pourrait dépasser la valeur max sur 16 bits (±32767), et vous verriez très vite une bien plus forte distorsion !

7.3.1. Vérification rapide du fonctionnement du programme.

Vérifier seulement le gain de 1, vers 1kHz.

Le déphasage n'est pas exploitable directement, pour deux raisons :

Le travail se fait en réalité par bloc, il y a donc un retard important entre l'entrée et la sortie et donc un déphasage proportionnel à la fréquence.

Les filtres (anti repliement et lissage) créent aussi leur propre déphasage (dus au retard et au filtrage) .

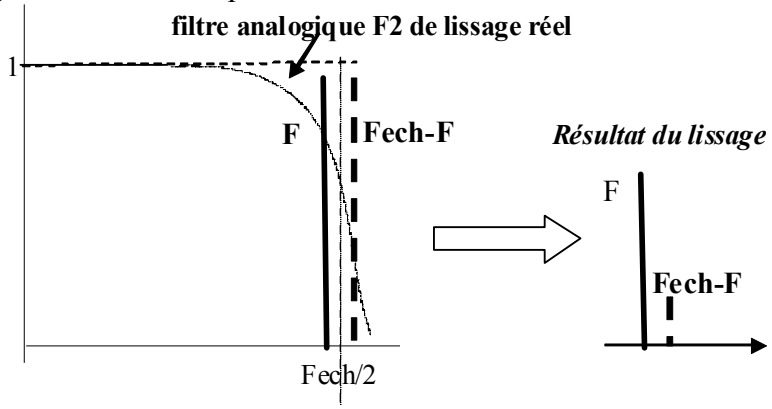
7.3.2. Relevé du module de gain de l'ensemble : échantillonneur et filtre de lissage, pour f <= Fech/2 (<=24 kHz)

7.3.2.1. Remarques théoriques

1) ***Lorsque F augmente*** et du fait de l'échantillonnage avec maintient (voir cours théorique), on devra observer petit à petit une légère atténuation, et à un moment donnée celle ci augmentera brusquement (intervention des filtres F1 et F2).

2) ***Lorsque F se rapproche de Fech/2*** et du fait du repliement de spectre, F et Fech-F deviennent proches. Deux cas peuvent se présenter :

→ **Si F et Fech-F ne sont pas trop atténuées (**par respectivement les **filtres F1 et F2 du cas B),** il reste F un peu atténué et Fech-F atténué d'avantage. On pourra visualiser alors nettement le mélange de ces deux fréquences.



Or, avec $\omega = 2\pi F$ et $\Omega = 2\pi(Fech - F)$, et en posant φ comme étant le déphasage entre les deux fréquences (dû au filtre), le signal filtré peut s'écrire :

$a.\sin \omega t + b.\sin(\Omega t + \varphi)$. Si on poursuivait les calculs on trouverait la somme de :

Un signal de fréquence $\omega = 2\pi F$

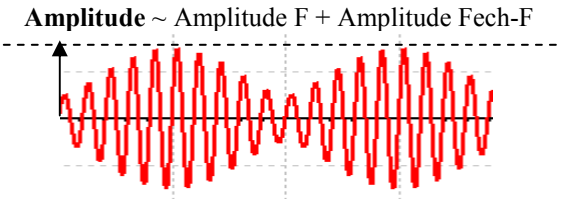
Un signal de fréquence $\Omega = 2\pi(Fech - F)$

Un signal du type $2a.\sin(\frac{\omega + \Omega}{2}t).\sin(\frac{\omega - \Omega}{2}t + \theta)$, et donc le produit d'une fréquence haute : $\omega + \Omega = Fech$ et d'une fréquence basse : $\omega - \Omega = 2F - Fech$

Tout ceci ressemble un peu à un signal modulé en amplitude : avec entre autre du **Fech** (qui serait la **porteuse**) modulé par du **2F - Fech** (qui serait le signal utile **BF**.

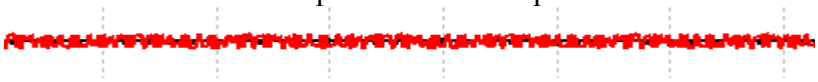
Quand on s'approche de Fech/2 (ou bien quand on le dépasse légèrement, on pourra donc observer une sorte de signal modulé AM (Le **battement est TBF** si **F est très voisin de Fech/2**).

On pourra juste **estimer l'ordre de grandeur de l'atténuation** du filtre.



Enveloppe = 2F-Fech = Bass fréquence

→**Si ces fréquences sont très atténuées (filtres F1 et F2 du cas A),** on retrouve ce même signal très faible et mélangé à du bruit, on ne peut rien en tirer ni mesurer vraiment une atténuation, on peut dire éventuellement que celle ci est supérieure à une certaine valeur.



7.3.2.2. Mesures pratiques

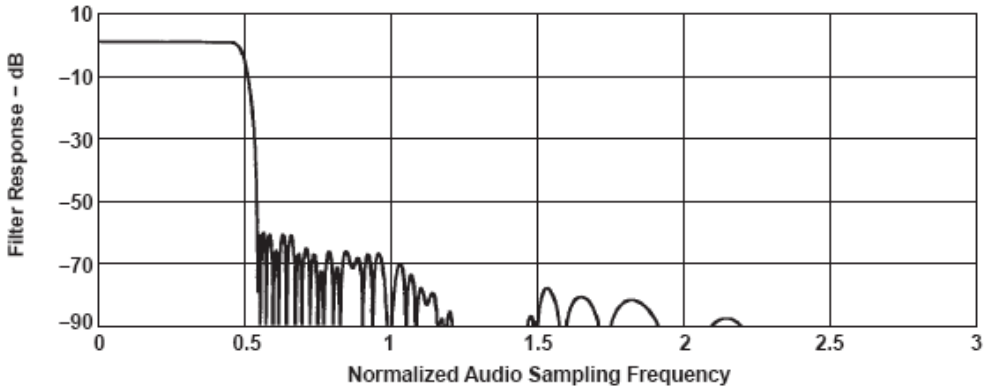
1) Mesure rapide du module de la fonction de transfert

On fera les mesures avec un générateur sinusoïdal réglé sur **environ 500 mVolt Crète**. Et on visualisera sur l'oscilloscope x(t) et y(t). En faisant varier la fréquence de 0 à Fech/2, observer et tracer (en échelles linaires) l'allure de la courbe de réponse en fréquence (module du gain directement, **non converti en dB**).

On connaît ici à priori la fonction de transfert que l'on doit vérifier (voir caractéristiques des filtres F1 et F2 fournies plus haut). Donc **ne pas mesurer inutilement des centaines de points n'importe où** ! **quelques mesures** bien placées sont suffisantes: une ou deux **en basse fréquence** : **10Hz**, **50Hz**, une **au milieu** de la bande passante, puis aux fréquences théoriques **Fmax= $F_{-0.05db}$** et proche de **F_{-6db}** trouvées précédemment sans dépasser pour l'instant Fech/2 = 24 kHz.

2) Comparaison avec la théorie

On redonne ici les caractéristiques théoriques de F1 et F2 (ici en dB). Vous avez déjà calculées dans la partie théorique les fréquences importantes dans votre cas.



- On peut déjà observer une différence importante en BF (< 50Hz), laquelle, expliquez d'où cela peut-t-il provenir.
- Comparez maintenant toutes vos mesures aux valeurs que vous avez déduites de la doc constructeur. Conclure.

3) « Shannon théorique » : rappeler la condition de Shannon pour pouvoir reconstituer et retrouver le signal de départ. Valeur numérique ?

4) « Shannon pratique » : estimer en pratique sur cette maquette la bande de fréquence (en Hz et de 0 à ?) sur laquelle vous aller pourrez réellement travailler, c'est à dire où l'ensemble « échantillonnage et restitution » peut être considérée comme sans influence (peut être à l'atténuation près due au principe même de l'échantillonneur bloqueur, que l'on pourrait d'ailleurs corriger assez aisément par un petit filtre si nécessaire).

7.3.3. Observation pour f > Fech/2 (f > 24 kHz)

1) Continuer la courbe précédente en relevant le module du gain pour 25kHz, 27kHz et 30 kHz par exemple.
2) Conclure en l'expliquant sur la présence ou non du filtre F1 anti repliement.

7.3.4. Mesure de la fréquence d'échantillonnage exacte, Fech

Il y a deux méthodes plus ou moins applicables selon les cas :

Mesure à F ~ Fech : On observe le battement de fréquence nulle (battement zéro) pour F = Fech. Pour cela, il ne faut évidemment ne pas avoir à l'entrée de filtre anti-repliement. On peut mesurer ainsi Fech à 1 Hz près si on veut !

Mesure à F ~ Fech/2 : Si l'atténuation n'est pas trop importante à Fech/2, on peut observer aisément le signal de sortie qui ressemble à une modulation d'amplitude, de fréquence de modulation très faible. On mesure ainsi Fech en trouvant le « **battement presque zéro** », à 1 Hz près si on veut !

1) D'après vos résultats précédents, sur votre système, quelle méthode allez vous pouvoir utiliser (en l'expliquant) ?
2) Effectuer cette mesure à quelques Hz près.

7.3.5. Meilleure observation du repliement de spectre, pour f > Fech/2

Comme il **y a toujours sur votre maquette** à l'entrée **le filtre F1** « anti-repliement » qui limite la bande, on ne peut évidemment pas sur cette maquette et ce programme observer le repliement de spectre, pour F > Fech/2. Mais on peut ruser !

1) Enlevez demo.c de votre projet et le remplacer par **demo_sous_ech.c**

La fonction de traitement est ainsi remplacée par :

```
void traitement_echantillons(int *I, int *O)
{
static char k=0;   static float ech=0;   // pour conserver un échantilon lu
// sous échantillonnage rapport 4:   Fech entrée = 48/4 = 12 kHz
if(k++>=3) { ech = (float)I[1]; k = 0;}
O[1] = fir(ech);   // voie droite, sortie lissée  et lissage à 6 kHz
O[0] = ech;                // sortie non lissée
}
```

Fech est toujours de 48kHz, donc les filtres d'anti-repliement et de lissage interviennent toujours à 24kHz.

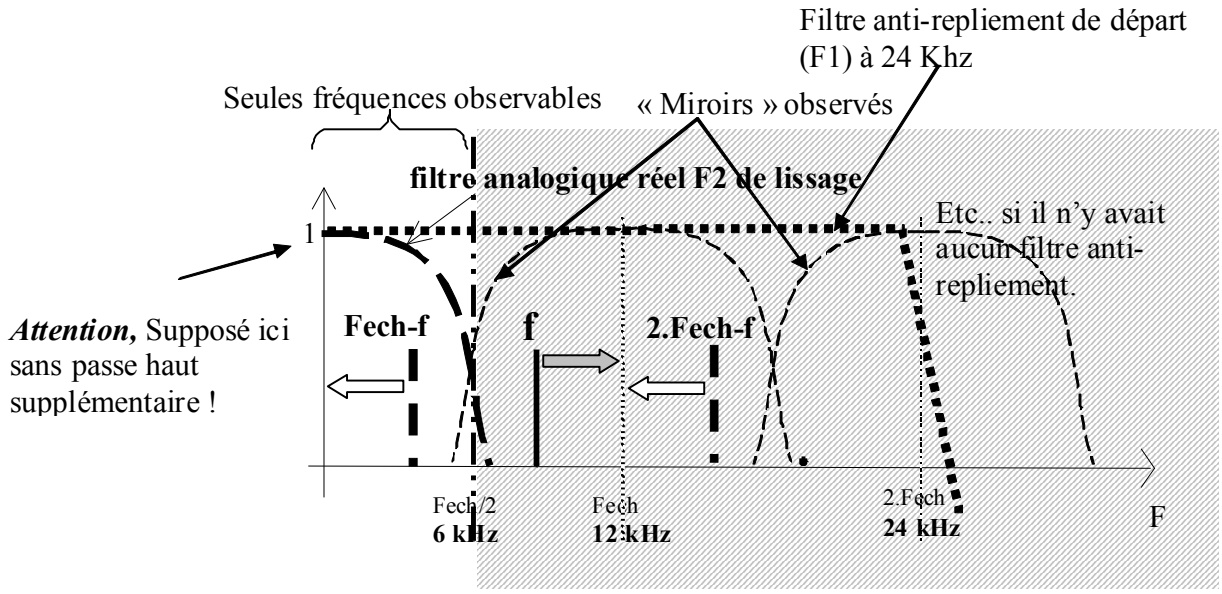
On a sous échantillonné dans un rapport 4, et on a ajouté sur la voie de droite un filtre de lissage numérique (FIR) à 6kHz (avec forte atténuation à 6kHz), coeff non donnés ici. Ne rien modifier !

On a donc : Nouvelle **Fech = 12 kHz**
Nouvelle **Fech/2 = 6kHz**
Nouveau **filtre de lissage en sortie, (F2) à 6kHz (forte atténuation à 6kHz)**
Le Filtre anti repliement (**F1)** à l'entrée interviendra seulement à 2*Fech = 24 kHz

2) En faisant varier la fréquence f du signal d'entrée de presque **0 à presque 2*Fech (ici 24KHz) tracer rapidement** (en ne mesurer que quelques points utiles !!!) le module de la fonction de transfert.

Pour F augmentant au-delà de Fech/2, on doit observer le « miroir » de la fonction de transfert précédente, ceci s'explique en fait par le « repliement de spectre » à l'entrée. La composante Fech – F du signal échantillonné xn évolue de Fech/2 à 0 (se déplace vers la gauche) et se retrouve ainsi dans le gabarit du filtre de lissage précédent. Ce n'est plus une vrai fonction de transfert, mais on peut parler de « pseudo fonction de transfert » et mesurer tout de même un module même si les fréquences à l'entrée et à la sortie ne sont plus les mêmes !

Au-delà de 24 kHz, on pourrait encore observer des miroirs, mais on ne verra ici plus rien le filtre anti repliement de départ (F1) étant toujours présent à 24kHz !!!



3) Pour f ≈ Fech, Fech-f ≈ 0, **observer un battement très basse fréquence** de grande amplitude. Par cette méthode, on pourrait mesurer précisément la nouvelle fréquence d'échantillonnage à quelques Hz près, non demandé.

4) Conclure sur la **nécessité** dans de nombreuses applications d'un filtre anti–repliement. On en décrira brièvement ses caractéristiques (pour être un bon filtre) en le comparant à celui du filtre de lissage. Bien comprendre que sans ce filtre, des fréquences assez élevées (du bruit par exemple) pourraient se retrouver en basse fréquence et distordre notablement notre signal utile.

Quelques applications utilisent néanmoins le « sous échantillonnage » sans filtre anti-repliement, afin de créer volontairement des changements de fréquences, par exemple pour des récepteurs de radio tout numérique.

8. TRAVAIL THEORIQUE ET PRATIQUE FILTRE MOYENNEUR FIR

La fréquence d'échantillonnage sera toujours fixée à :
Fech ≈ 48 KHz

→ **Pour un DSP virgule flottante (comme notre C67) :**
Les échantillons entiers deviennent des flottants (par cast implicite en C).
Les coefficient sont évidemment aussi en flottant.

Tous les calculs se font en flottant.
On ne peut donc déborder que lors d'une sortie éventuelle sur un CNA. D'ou l'importance d'une réflexion préalable sur la dynamique pratique de sortie.

8.1. Dynamique pratique à la sortie d'un traitement quelconque

Les **échantillons** d'entrée et sortie sont des **valeurs entières** (**entiers** ou **flottants**) de **– 32768** à **32767**, correspondant à des **tensions d'entrées** de **-1,3V** à **presque +1,3V**.
Mais il peut être parfois plus simple de raisonner en Q15(16), c'est à dire avec un bit de partie entière et 15 fractionnaires, et donc avec des nombres en virgule fixe **par la pensée** de **module < 1**. Format : -,------- --------

Raisonnons donc ici avec **x de module < 1**, au format **Q15(16) : -,------- --------**
On peut distinguer trois cas principaux :

→ ***Cas N°1, le plus courant :*** la réponse indicielle (régime transitoire) tout comme la réponse fréquentielle peut être oscillatoire et donc présenter un certain dépassement, on peut alors prévoir une sortie **y** de dynamique double, format Q14(16) : --,------ --------

Pour ne pas déborder au CNA (qui reçoit du Q15(16)), il faudra donc lui envoyer $\mathbf{1/2\ y_n}$ soit la moitié de la convolution dans ce cas général.

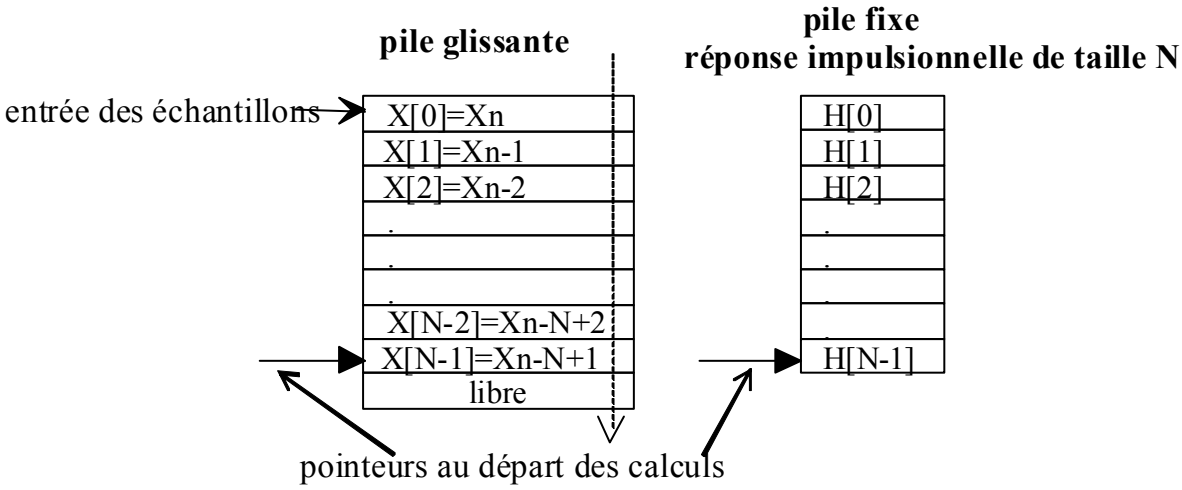
→ ***Cas N°2 :*** filtres très oscillants, ou très sélectifs, les corrélations également, demanderont bien plus de dynamique, on sortira alors $\mathbf{1/2^k\ y_n}$**.**

→ ***Cas N°3 :*** la sortie du filtre y_n reste toujours (même en transitoire !) de module < 1, on peut alors garder le même format que les x_n, et sortir donc $\mathbf{y_n}$.
(Ce sera le cas par exemples de filtres réalisant une simple moyenne)

Remarques utiles: raisonnement | xn | <=1

- Valeur finale de y pour la réponse indicielle, et **gain pour f = 0** : $\sum_{k=0}^{N-1} h_k$
- **Valeur max** de y **cas le plus défavorable**: $\sum_{k=0}^{N-1} |h_k|$
- Valeur max de y lors d'un transitoire, il faudrait l'étudier, mais on peut prendre le cas le plus défavorable précédent.
- Attention : même si le module du gain d'un filtre est toujours < 1, la réponse en transitoire peut osciller et présenter des dépassements au delà de 1.

8.2. Principe de calcul, avec pile glissante



La case mémoire après X[N-1] doit être libre !!, car les données se décalent à chaque fois d'un cran vers le bas et donc elles passent toutes par la dernière case Dans le cas contraire, on risquerait fort d écraser sans le savoir une valeur importante, qui se trouverait par hasard à cet endroit ! (En programmation, il faut toujours savoir ce que l'on fait, sinon sans précaution, un programme peut marcher bien un certain temps, puis d'un seul coup après une modification n'ayant rien à voir avec ces lignes, se dégrader ou faire n'importe quoi, et paraître hanté …. !)

Une somme de produits doit donc se calculer après chaque acquisition xn pour calculer yn.

Le calcul **démarre** par la **fin du tableau** pour pouvoir à chaque boucle décaler la valeur pointée vers le bas. Toute la pile X est ainsi décalée d'un cran à la fin de chaque calcul de yn.

8.3. Examen du programme fourni : filtre1.c Moyenne mobile sur 21 échantillons

→ Dans le projet **dsk_c67.pjt** du **« Code Composeur Studio »** (qui est l'outil de développement fourni par Texas),

Aller **sources**, il vous **faudra** maintenant les fichiers C principaux (et seulement ceux ci) :

dsk_c67.c	le noyau de l'application, ne rien modifier !
aic23.c	fonctions de l'interface analogique, ne rien modifier
filtre1.c	**Le fichier C de travail** (on devra donc retirer du projet demo.c ou demo_sous_ech.c, et le remplacer par filtre1.c)

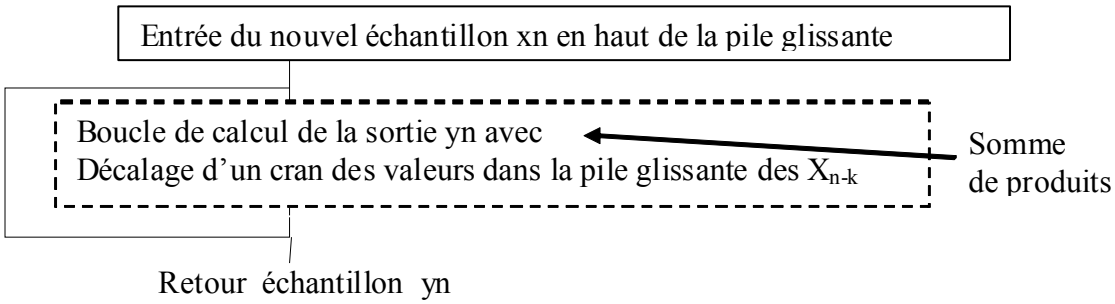
On peut distinguer dans filtre1.c

- **Un #define taille …** // taille du filtre
- **Les deux tableaux en variable globale :**
  - **float H[taille];** // réponse impulsionnelle de taille : **taille**.
  - **float X[taille+1];** // la pile glissante d'échantillons, de taille : **taille+1**

- **La fonction : Fonction init_traitement()** // Initialisation des coefficients

- ***La Fonction*** **void traitement_echantillons(int *I, int *O)**
  Appel de la fonction float fir1(float xn) ; avec l'entrée I[1] voie droite, renvoi vers la sortie O[1] voie droite.
- ***La Fonction*** **float fir1(float xn) ;** // calcul de la convolution, le filtre FIR

FONCTION FILTRE NUMERIQUE FIR



ATTENTION : Ne cherchez toujours pas de fonction main() !!!! Il se trouve dans un autre fichier.

- ***Programme complet fourni :*** fichier ***filtre1.C***

```
//                      PROJET toujours dsk_c6713.pjt
// kit de développement TEXAS TMS 320C6713 DSK
// I[0] I[0]:  voie gauche ---> "tip" ; fiche RCA blanche ou noire
// O[1] O[1]: voie DROITE ---> "ring" ; fiche RCA ROUGE
// Dynamique des échantillons:
//      +- 1,3 Volt crète
//      -32767 à +32767    en Q0(16)        (sur des int de 32 bits)
//      presque (-1 à +1)  en Q15(16)  ( Q31(32) en fait )
//Si on veut travailler sur des flottants de -1 à +1,
//              On divisera par 32768.0  en entrée
//              On re multipliera par 32768.0 avant de sortir
// @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
#include <log.h>
#include <math.h>
extern far LOG_Obj trace;   // si on voulait des message log
void pulse_gpio8(void);   // pour mesurer des durées en temps réel

//   DONNEES POUR LE FILTRE FIR
#define taille 21           // ici 21 coefficients
float H[taille];            // réponse impulsionnelle
float X[taille+1];          // Pile glissante des x, de  taille+1 obligatoire !
float fir(float xn);

float fir(float xn)
{
int k;
float yn=0;
X[0] = xn;   // entrée échantillon dans pile glissante x[]
     for(k=taille-1;k>=0;k--)       // CONVOLUTION
               {        yn+=X[k]*H[k];        // (somme des produits)
                        X[k+1] = X[k];                  // (glissement)
               }
return(yn);     // retour échantillon calculé
}
```

```
// Initialisations avant la boucle principale de travail
Init_traitement()          // Initialisation des coefficients
{
int k;
for(k=0; k < taille+1; k++)X[k] = 0;     // utile en debug

//  Initialisation des coefficients :
for(k=0; k < taille+1; k++)H[k] = 1.0/taille;   // Coefficients initialisés ici à 1/taille
}

// Traitement des échantillons comme une simple acquisition un par un (2 voies)
void traitement_echantillons(int *I, int *O)
{//    VOIE 0:   [0]      VOIE 1:   [1]              ici on travaille sur la voie 1
float yn;
yn = fir(I[1]);                // filtrage voie 1
O[1] = yn;           // sortie voie 1 non divisée par 2,  cas N°3 (étude des débordements)
}

void periodic_log(void)
{ }  // pour affichage de résultats par fonction  LOG_printf(…) ;  Voir plus loin.
```

→ Explications des calculs de la fonction fir:

Les échantillons I et O de la fonction traitement échantillons sont:

Des entiers de -32767 à +32768,

Ou des valeurs de –1 à presque +1 en raisonnement par la pensée en virgule fixe Q15(16) **-,------- --------**

Ces valeurs sont automatiquement converties en flottants par des cast implicites (tableau des X est déclaré en flottant, elles valent donc ensuite réellement en flottant toute valeurs entières de –32768 à +32767).

Les coefficients H (réponse impulsionnelle) sont en flottant. Tous les calculs se font en flottant, donc pas de soucis de débordement durant ceux ci.

Pour sortir sur le CNA, seule la dynamique pratique compte, comme nous l'avons déjà expliqué. Ici on sort directement la convolution (et non ½) pour ce petit filtre tout simple (pas de débordement).

On ne risque évidemment **pas de déborder** en calculant une moyenne, qui sera toujours plus petite ou égale au plus grand, on reste donc dans le cas N°3 où l'on ne réduit pas le signal de sortie. On sort donc ici directement yn.

8.4. Préparation théorique : filtre de taille 21

1) Dessinez **rapidement** le module **théorique** de la fonction de transfert pour F variant de 0 à Fech/2 (gain pour F faible et valeurs des 2 premiers 0).
2) Donner les **fréquences** exactes des 2 premiers zéros.

8.5. Travail pratique : gain et régime transitoire

8.5.1. Fonction de tranfert, pour F < Fech/2

1) Vérifier pratiquement le module (de F faible, jusqu'au deux premiers zéros) de la fonction de transfert du filtre obtenu, avec toujours la bonne mesure de quelques points caractéristiques : en BF (10 et 50Hz), et un ou deux points supplémentaires avant le **premier zéro** qui doit être **exactement à Fech/n**. On comparera **très pertinemment** théorie et pratique, si vous constater des différences (en particulier en BF en dessous de 30Hz par exemple, les expliquer correctement ! (voir carte utilisée et filtres existant).
2) Qu'observe-t-on quand F s'approche de Fech/2. Expliquez.

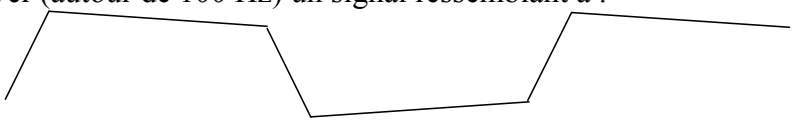
3) Rappeler quel est le point le plus important à contrôler (que vous avez du vérifié !) afin d'être sûr de la bonne taille de filtre, et que le temps réel est bien assuré à cette taille (sinon ce point ne serait plus bon sans parler d'autres perturbations possibles).

8.5.2. Régime transitoire

1) On sait que la durée du régime transitoire d'un filtre numérique est égal à la durée de la réponse impulsionnelle. Calculez cette durée pour ce filtre.

2) Vérifier cette durée en pratique en observant la réponse à un signal carré (de fréquence permettant évidemment une bonne mesure !). On **indiquera** bien **où se situe** ce régime transitoire, et on en mesurera **la durée**.

On doit observer (autour de 100 Hz) un signal ressemblant à :



3) Les zones qui devraient horizontales sur votre signal carré ne le sont pas vraiment, expliquer pourquoi.

9. THEORIE ET PRATIQUE: MESURE DE VALEUR EFFICACE

9.1. Modification du programme

Comme notre maquette DSK C67 comporte hélas une liaison capacitive en entrée, on ne peut évidemment pas mesurer une valeur moyenne qui serait toujours nulle ! On mesurera donc une valeur efficace.

$$Veff = \sqrt{\frac{1}{N}\sum_{0}^{N-1} x_n^2}$$

D'autre part, la maquette comporte aussi hélas une liaison capacitive en sortie, on ne pourra donc pas **visualisez** une valeur constante en sortie, mais **seulement les fluctuations** de la moyenne lorsque nous ne somme pas dans les conditions d'une bonne mesure.

Pour contrôler la bonne moyenne, on utilisera la fonction **void periodic_log(void)** du système qui est exécutée automatiquement (vous avez en réalité un petit système multi-tâches) toutes les quelques secondes.

Il est possible par une fonction système **LOG_printf** d'afficher texte et valeurs dans une fenêtre de votre système de DEBUG (l'opérande est écrit comme pour les printf du C).

Pour afficher **Veff**, cette variable devra être **en global obligatoirement**.

1) Modification de votre programme
   Faire la moyenne mobile de non plus xn mais xn*xn
   En déduire Veff
   Pour l'affichage (fonction **LOG_printf(..)**) de **Veff** en **Volt** dans le processus **void periodic_log(void),** sachant que le **CAN** travaille sur une tension d'entrée de **1,3v crête (-1.3v à presque +1.3v**), pour des échantillons sur **16 bits signés (-32768 à 32767)**, on complétera correctement la ligne correspondante aux ??????.

```
float veff;            // en global
void traitement_echantillons(int *I, int *O)
{//    VOIE GAUCHE: [0]      VOIE DROITE: [1]  fiche rouge
??????????
veff =  ??????????
O[1] = veff;                  // sortie voie Droite
}

void periodic_log(void)
{
LOG_printf(&trace, "Veff = %f   v",   ?????????? );     // Affichage de Veff en volt
}
```

2) Pourquoi faut-il mettre Veff en variable globale ?

3) Vérification pratique :

La visualisation de Veff se fait en réalité dans une fenêtre à ouvrir avec :
DSP/BIOS | Message Log
Puis faire **click droit** et **Property page** et **cocher** 'Automatically scroll end of buffer'.

Rappel :
Dans les conditions d'une bonne mesure :
Le signal à l'oscillo sera pratiquement 0 (pas de variations).
Les tensions affichées Veff seront stables.
Dans de mauvaises conditions :
Le signal à l'oscillo fluctue.
Les tensions affichées Veff seront instables.

On fera les essais suivants, et **pour chaque cas** on **décrira**, et on **dessinera** rapidement ce que l'on observe à l'oscilloscope, et on **vérifiera évidemment** la bonne mesure de Veff !

- 🕮 En envoyant un signal sinusoïdal de fréquence F0 et d'amplitude A0 (< 1.3v) adéquates, vérifier pratiquement la bonne mesure de la valeur efficace dans les deux cas étudiés plus haut :
- Cas particulier ou F0 = Fech/N ou 2.Fech/N ..
- Cas général ou la fréquence n'est pas connue, il faut alors seulement : F0 >> Fech/N

- 🕮 Se placer ensuite dans le cas d'une mauvaise mesure.

- 🕮 Fixer la taille à par exemple **1400** (on suppose que le temps réel est encore respecté). Vérifier à l'œil l'amélioration nette des mesures dans le cas général.

9.2. Calcul précis du nombre de points de mesure

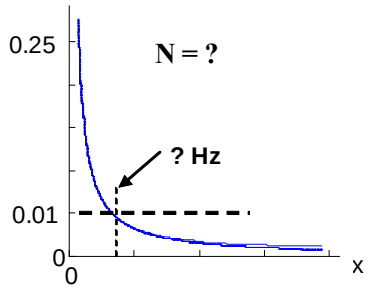
Cahier des charges :
Échantillonnage à Fech = 48 KHz (comme précédemment).
Obtention de la valeur efficace avec pour **F ≥ 50 Hz ondulations inférieures à 1%**

9.2.1. Utilisation de notre filtre FIR Moyenneur

1) Sachant qu'on filtre non pas le signal directement, mais son carré, quelle sera la fréquence minimale à prendre en compte ?
(se rappeler de la formule de trigonométrie par exemple $\sin^2 x = \frac{1-\cos 2x}{2}$)

2) Calculer la taille de filtre nécessaire, on utilisera l'équation de la courbe des maximums et on fera sinx ≈ x en début de courbe.

Courbe des maximums du module du gain



3) Est-ce possible avec notre programme précédent, si on suppose une taille maximale possible avec ce DSP de 1800 ?
4) Conclusion :

On constate bien la limite de notre DSP programmé en C, car nous avons déjà en option de compilation l'optimisation maximum du code généré.

- Des **optimisations en vitesse** d'exécution sont **possibles** en écrivant en assembleur des fonctions C, pour exploiter totalement la puissance de certaines instructions du DSP. On peut gagner parfois dans un rapport 5 ou plus ! mais ce n'est pas le but de ces TP. Les compilateurs C sont de nos jours de plus en plus puissant et génèrent déjà un code très optimisé. D'autre part, la programmation des DSP actuels en assembleur est de plus en plus ardue, avec parfois 4 instructions ou plus pouvant être exécutées en même temps!).
- Les constructeurs donnent des **librairies de fonctions C écrites en assembleurs**, on peut les utiliser alors directement,

Pour de simple moyennes mobiles, il existerait en fait un **autre type de filtre moyenneur**, le moyenneur **récursif** (par filtre IIR) qui nécessiterait bien moins d'opération ! Non étudié ici.

9.2.2. Autre méthode, par bloc, sans véritable filtre !

Il est évident que pour mesurer une simple valeur moyenne ou efficace, il est inutile d'avoir en permanence le résultat, on peut n'**actualiser** par exemple **que toutes les secondes** le résultat. Un filtre n'est alors pas utile, il suffit de faire un simple calcul par bloc : acquérir N échantillons puis ensuite seulement calculer la Veff.

Par exemple, avec une mesure sur **1 seconde à 48kHz**, on aurait **48000 échantillons**, on améliorerait alors grandement la mesure (l'ondulation passerait à 0,3% ou bien on pourrait descendre bien plus bas en fréquence minimale en gardant les 1%).

→On peut obtenir ainsi la précision que l'on souhaite au détriment évidemment de la durée totale de mesure.

PAS DE PARTIE PRATIQUE